ScheduledRecording:2 Service Template Version 1.01

For UPnP™ Version 1.0
Status: Standardized DCP
Date: September 30, 2008
Document Version: 1.0

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

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

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
### Authors*

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

*Note: The UPnP Forum in no way guarantees the accuracy or completeness of this author list and in no way implies any rights for or support from those members listed. This list is not the specifications’ contributor list that is kept on the UPnP Forum’s website.
Contents

1 Overview and Scope .......................................................................................................................... 13
   1.1 Introduction ................................................................................................................................ 13
   1.2 Notation ........................................................................................................................................ 14
       1.2.1 Data Types ........................................................................................................................... 14
       1.2.2 Strings Embedded in Other Strings ..................................................................................... 14
       1.2.3 Extended Backus-Naur Form ............................................................................................... 15
   1.3 Derived Data Types ...................................................................................................................... 15
       1.3.1 Comma Separated Value (CSV) Lists .................................................................................. 16
   1.4 Management of XML Namespaces in Standardized DCPs ............................................................ 17
       1.4.1 Namespace Prefix Requirements ......................................................................................... 20
       1.4.2 Namespace Names, Namespace Versioning and Schema Versioning .................................. 21
       1.4.3 Namespace Usage Examples ............................................................................................... 22
   1.5 Vendor-defined Extensions .......................................................................................................... 23
       1.5.1 Vendor-defined Action Names ............................................................................................. 23
       1.5.2 Vendor-defined State Variable Names .................................................................................. 23
       1.5.3 Vendor-defined XML Elements and attributes ....................................................................... 23
       1.5.4 Vendor-defined Property Names ........................................................................................ 24
   1.6 References ................................................................................................................................... 24

2 Service Modeling Definitions ........................................................................................................... 28
   2.1 ServiceType .................................................................................................................................. 28
   2.2 Terms and Abbreviations ............................................................................................................. 28
       2.2.1 Abbreviations ....................................................................................................................... 28
       2.2.2 Terms .................................................................................................................................... 28
   2.3 ScheduledRecording Service Architecture .................................................................................. 34
       2.3.1 recordSchedule .................................................................................................................... 34
       2.3.2 recordTask ............................................................................................................................ 36
   2.4 State Variables ............................................................................................................................. 36
       2.4.1 State Variable Overview ......................................................................................................... 37
       2.4.2 SortCapabilities .................................................................................................................... 38
       2.4.3 SortLevelCapability .............................................................................................................. 38
       2.4.4 StateUpdateID ....................................................................................................................... 38
       2.4.5 LastChange ........................................................................................................................... 39
       2.4.6 A_ARG_TYPE_PropertyList ................................................................................................. 41
       2.4.7 A_ARG_TYPE_DataTypeID .................................................................................................. 42
       2.4.8 A_ARG_TYPE_ObjectID ........................................................................................................ 42
       2.4.9 A_ARG_TYPE_ObjectIDList ................................................................................................ 42
       2.4.10 A_ARG_TYPE_PropertyInfo ................................................................................................. 42
       2.4.11 A_ARG_TYPE_Index ............................................................................................................... 42
       2.4.12 A_ARG_TYPE_Count ............................................................................................................ 42
       2.4.13 A_ARG_TYPE_SortCriteria ................................................................................................. 42
       2.4.14 A_ARG_TYPE_RecordSchedule ........................................................................................... 43
       2.4.15 A_ARG_TYPE_RecordTask ................................................................................................... 43
2.4.16  A_ARG_TYPE_RecordScheduleParts ................................................................. 43
2.5  Eventing and Moderation ...................................................................................... 44
2.6  Actions .................................................................................................................. 44
  2.6.1  GetSortCapabilities() ..................................................................................... 45
  2.6.2  GetPropertyList() .......................................................................................... 46
  2.6.3  GetAllowedValues() ........................................................................................ 47
  2.6.4  GetStateUpdateID() ....................................................................................... 49
  2.6.5  BrowseRecordSchedules() ............................................................................... 50
  2.6.6  BrowseRecordTasks() ...................................................................................... 54
  2.6.7  CreateRecordSchedule() .................................................................................. 56
  2.6.8  DeleteRecordSchedule() ................................................................................... 58
  2.6.9  GetRecordSchedule() ....................................................................................... 59
  2.6.10 EnableRecordSchedule() ................................................................................. 60
  2.6.11 DisableRecordSchedule() ............................................................................... 61
  2.6.12 DeleteRecordTask() ........................................................................................ 63
  2.6.13 GetRecordTask() ............................................................................................. 63
  2.6.14 EnableRecordTask() ........................................................................................ 64
  2.6.15 DisableRecordTask() ....................................................................................... 66
  2.6.16 ResetRecordTask() .......................................................................................... 67
  2.6.17 GetRecordScheduleConflicts() ........................................................................ 68
  2.6.18 GetRecordTaskConflicts() .............................................................................. 69
  2.6.19 Common Error Codes ...................................................................................... 70
2.7  State Diagram of recordTask .................................................................................. 71
  2.7.1  A Full-Featured State Diagram ......................................................................... 72
  2.7.2  A Minimal-Implementation State Diagram ...................................................... 77
  2.7.3  recordTask State Example ............................................................................... 80
2.8  ScheduledRecording Service Priority Model .............................................................. 81
  2.8.1  Introduction of the ScheduledRecording Service Priority Model ...................... 81
  2.8.2  Ordered Priority within Each Priority Level ...................................................... 82
  2.8.3  Setting the Initial Priority Level of a recordSchedule ........................................ 83
  2.8.4  Sorting recordSchedule Instances Based on their Current Priority Settings ....... 85
2.9  Theory of Operation ............................................................................................... 85
  2.9.1  Introduction ..................................................................................................... 85
  2.9.2  Checking the Capabilities of a ScheduledRecording Service ............................ 86
  2.9.3  Adding a Scheduled Recording Entry to the List .............................................. 96
  2.9.4  Deleting a recordSchedule .............................................................................. 110
  2.9.5  Browsing recordSchedule and recordTask instances .................................... 110
  2.9.6  Rating System .................................................................................................. 116
  2.9.7  Conflict Detection and Resolution ................................................................... 117
3  XML Service Description ............................................................................................ 119
4  Test ........................................................................................................................... 129
Appendix A.  srs XML Document (Normative) ................................................................. 130
  A.1  A_ARG_TYPE_RecordSchedule AVDT XML Document ........................................ 130
### Appendix B. AV Working Committee Extended Properties (Normative)

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1</td>
<td>Base Properties</td>
<td>133</td>
</tr>
<tr>
<td>B.1.1</td>
<td>@id</td>
<td>133</td>
</tr>
<tr>
<td>B.1.2</td>
<td>title</td>
<td>133</td>
</tr>
<tr>
<td>B.1.3</td>
<td>class</td>
<td>134</td>
</tr>
<tr>
<td>B.1.4</td>
<td>additionalStatusInfo</td>
<td>134</td>
</tr>
<tr>
<td>B.1.5</td>
<td>cdsReference</td>
<td>135</td>
</tr>
<tr>
<td>B.2</td>
<td>Priority Properties</td>
<td>135</td>
</tr>
<tr>
<td>B.2.1</td>
<td>priority</td>
<td>136</td>
</tr>
<tr>
<td>B.2.2</td>
<td>desiredPriority</td>
<td>137</td>
</tr>
<tr>
<td>B.2.3</td>
<td>desiredPriority@type</td>
<td>138</td>
</tr>
<tr>
<td>B.3</td>
<td>Output Control Properties</td>
<td>139</td>
</tr>
<tr>
<td>B.3.1</td>
<td>recordDestination</td>
<td>139</td>
</tr>
<tr>
<td>B.3.2</td>
<td>desiredRecordQuality</td>
<td>141</td>
</tr>
<tr>
<td>B.4</td>
<td>Content Identification Related Properties</td>
<td>144</td>
</tr>
<tr>
<td>B.4.1</td>
<td>scheduledCDSObjectID</td>
<td>144</td>
</tr>
<tr>
<td>B.4.2</td>
<td>scheduledChannelID</td>
<td>145</td>
</tr>
<tr>
<td>B.4.3</td>
<td>scheduledStartDateTime</td>
<td>147</td>
</tr>
<tr>
<td>B.4.4</td>
<td>scheduledDuration</td>
<td>147</td>
</tr>
<tr>
<td>B.4.5</td>
<td>scheduledProgramCode</td>
<td>148</td>
</tr>
<tr>
<td>B.5</td>
<td>Matching Content Criteria Properties</td>
<td>148</td>
</tr>
<tr>
<td>B.5.1</td>
<td>matchingName</td>
<td>148</td>
</tr>
<tr>
<td>B.5.2</td>
<td>matchingID</td>
<td>149</td>
</tr>
<tr>
<td>B.6</td>
<td>Matching Qualifying Criteria Properties</td>
<td>151</td>
</tr>
<tr>
<td>B.6.1</td>
<td>matchingChannelID</td>
<td>151</td>
</tr>
<tr>
<td>B.6.2</td>
<td>matchingStartDateTimeRange</td>
<td>152</td>
</tr>
<tr>
<td>B.6.3</td>
<td>matchingDurationRange</td>
<td>152</td>
</tr>
<tr>
<td>B.6.4</td>
<td>matchingRatingLimit</td>
<td>153</td>
</tr>
<tr>
<td>B.6.5</td>
<td>matchingEpisodeType</td>
<td>155</td>
</tr>
<tr>
<td>B.7</td>
<td>Content Control Properties</td>
<td>156</td>
</tr>
<tr>
<td>B.7.1</td>
<td>totalDesiredRecordTasks</td>
<td>156</td>
</tr>
<tr>
<td>B.7.2</td>
<td>scheduledStartDateTimeAdjust</td>
<td>157</td>
</tr>
<tr>
<td>B.7.3</td>
<td>scheduledDurationAdjust</td>
<td>157</td>
</tr>
<tr>
<td>B.7.4</td>
<td>activePeriod</td>
<td>157</td>
</tr>
<tr>
<td>B.7.5</td>
<td>durationLimit</td>
<td>158</td>
</tr>
<tr>
<td>B.7.6</td>
<td>channelMigration</td>
<td>159</td>
</tr>
<tr>
<td>B.7.7</td>
<td>timeMigration</td>
<td>159</td>
</tr>
<tr>
<td>B.7.8</td>
<td>allowDuplicates</td>
<td>159</td>
</tr>
<tr>
<td>B.8</td>
<td>Storage Related Properties</td>
<td>160</td>
</tr>
<tr>
<td>B.8.1</td>
<td>persistedRecordings</td>
<td>160</td>
</tr>
<tr>
<td>B.9</td>
<td>Schedule State Properties</td>
<td>161</td>
</tr>
<tr>
<td>B.9.1</td>
<td>scheduleState</td>
<td>161</td>
</tr>
<tr>
<td>Appendix C.</td>
<td>AV Working Committee Class Definitions (Normative)</td>
<td>188</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>C.1</td>
<td>Class Hierarchy</td>
<td>188</td>
</tr>
<tr>
<td>C.1.1</td>
<td>Relationships between Classes and Properties</td>
<td>189</td>
</tr>
<tr>
<td>C.1.2</td>
<td><code>recordScheduleParts</code> Properties</td>
<td>190</td>
</tr>
<tr>
<td>C.1.3</td>
<td><code>recordSchedule</code> Properties</td>
<td>194</td>
</tr>
<tr>
<td>C.1.4</td>
<td><code>recordTask</code> Properties</td>
<td>198</td>
</tr>
<tr>
<td>C.2</td>
<td>Class Definitions</td>
<td>202</td>
</tr>
<tr>
<td>C.3</td>
<td><code>object</code> Base Class</td>
<td>203</td>
</tr>
<tr>
<td>C.3.1</td>
<td><code>object.recordSchedule</code> Class</td>
<td>204</td>
</tr>
<tr>
<td>C.3.2</td>
<td><code>object.recordTask</code> Class</td>
<td>214</td>
</tr>
</tbody>
</table>

| Appendix D. | EBNF Syntax Definitions (Normative) | 216 |

---

B.10 Statistics Properties
B.10.1 `currentRecordTaskCount` ................................................................. 163
B.10.2 `totalCreatedRecordTasks` ............................................................... 163
B.10.3 `totalCompletedRecordTasks` ........................................................... 164
B.11 Task General Properties ....................................................................... 164
B.11.1 `recordScheduleID` ........................................................................... 164
B.11.2 `recordCDSObjectID` ......................................................................... 164
B.12 Task Content Identification Properties .............................................. 165
B.12.1 `taskCDSObjectID` .............................................................................. 165
B.12.2 `taskChannelID` .................................................................................. 166
B.12.3 `taskStartDate` .................................................................................. 167
B.12.4 `taskDuration` ................................................................................... 167
B.12.5 `taskProgramCode` ............................................................................ 168
B.12.6 `recordQuality` .................................................................................. 168
B.13 Task Matched Content Criteria Properties ......................................... 170
B.13.1 `matchedName` .................................................................................. 170
B.13.2 `matchedID` ....................................................................................... 171
B.14 Task Matched Qualifying Criteria Properties ...................................... 171
B.14.1 `matchedRating` .................................................................................. 171
B.14.2 `matchedRating@type` ...................................................................... 172
B.14.3 `matchedEpisodeType` ...................................................................... 172
B.15 Task Matched Content Control Properties ....................................... 172
B.15.1 `taskStartDateAdjust` ...................................................................... 173
B.15.2 `taskDurationAdjust` ........................................................................ 173
B.15.3 `taskDurationLimit` .......................................................................... 173
B.15.4 `taskDurationLimit@effect` ................................................................. 173
B.15.5 `taskChannelMigration` ................................................................... 174
B.15.6 `taskTimeMigration` ......................................................................... 174
B.16 Task State Properties .......................................................................... 174
B.16.1 `taskState` ......................................................................................... 175
B.17 ContentDirectory Service Imported Properties .................................. 183

---

Object Class Definitions ........................................................................ 202
Class Hierarchy .................................................................................... 203
ContentDirectory Service Imported Properties .................................... 188
Task Matched Content Identification Properties ..................................... 170
Task Matched Qualifying Criteria Properties ....................................... 171
Task Matched Content Control Properties ....................................... 172
Task State Properties .......................................................................... 174
Class Definitions .................................................................................. 202
object Base Class .................................................................................. 203
object.recordSchedule Class ............................................................... 204
object.recordTask Class ....................................................................... 214

---

EBNF Syntax Definitions (Normative) ...................................................... 216

---

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
D.1 Priority Syntax .............................................................. 216
D.2 Date&time Syntax ............................................................ 216
D.3 Class Name Syntax .......................................................... 217

Appendix E. ScheduledRecording Service Relationship to ContentDirectory Service (Informative) 218

Appendix F. ScheduledRecording Service Relationship to EPG (Informative) ............................. 219

Appendix G. AVDT Examples (Informative) .................................................................................... 220
G.1 A_ARG_TYPE_RecordSchedule AVDT Example ........................................................................ 220
G.2 A_ARG_TYPE_RecordTask AVDT Example ............................................................................. 238
G.3 A_ARG_TYPE_RecordScheduleParts AVDT Example .......................................................... 257
List of Tables

Table 1-1: EBNF Operators ............................................................................................................... 15
Table 1-2: CSV Examples .................................................................................................................. 16
Table 1-3: Namespace Definitions ..................................................................................................... 18
Table 1-4: Schema-related Information .............................................................................................. 19
Table 1-5: Default Namespaces for the AV Specifications ................................................................. 20
Table 2-1: Abbreviations .................................................................................................................... 28
Table 2-1: Properties in XML ............................................................................................................. 30
Table 2-2: State Variables .................................................................................................................. 37
Table 2-3: allowedValueList for the $DataTypeID$ argument .......................................................... 37
Table 2-4: Allowed Elements in $<StateEvent>$ Element ................................................................ 39
Table 2-5: Eventing and Moderation .................................................................................................. 44
Table 2-6: Actions ................................................................................................................................ 44
Table 2-7: Arguments for $GetSortCapabilities()$ ........................................................................... 46
Table 2-8: Error Codes for $GetSortCapabilities()$ ......................................................................... 46
Table 2-9: Arguments for $GetPropertyList()$ ..................................................................................... 47
Table 2-10: Error Codes for $GetPropertyList()$ ............................................................................... 47
Table 2-11: Arguments for $GetAllowedValues()$ ............................................................................ 48
Table 2-12: Error Codes for $GetAllowedValues()$ .......................................................................... 49
Table 2-13: Arguments for $GetStateUpdateID()$ ............................................................................. 49
Table 2-14: Error Codes for $GetStateUpdateID()$ ........................................................................... 49
Table 2-15: Arguments for $BrowseRecordSchedules()$ .................................................................. 50
Table 2-16: Error Codes for $BrowseRecordSchedules()$ ................................................................. 54
Table 2-17: Arguments for $BrowseRecordTasks()$ ......................................................................... 54
Table 2-18: Error Codes for $BrowseRecordTasks()$ ....................................................................... 55
Table 2-19: Arguments for $CreateRecordSchedule()$ ..................................................................... 56
Table 2-20: Error Codes for $CreateRecordSchedule()$ ................................................................. 57
Table 2-21: Arguments for $DeleteRecordSchedule()$ ...................................................................... 59
Table 2-22: Error Codes for $DeleteRecordSchedule()$ ..................................................................... 59
Table 2-23: Arguments for $GetRecordSchedule()$ ........................................................................ 59
Table 2-24: Error Codes for $GetRecordSchedule()$ ....................................................................... 60
Table 2-25: Arguments for $EnableRecordSchedule()$ ..................................................................... 61
Table 2-26: Error Codes for $EnableRecordSchedule()$ ................................................................. 61
Table 2-27: Arguments for $DisableRecordSchedule()$ .................................................................... 62
Table 2-28: Error Codes for $DisableRecordSchedule()$ ................................................................. 62
Table 2-29: Arguments for $DeleteRecordTask()$ .......................................................................... 63
Table 2-30: Error Codes for `DeleteRecordTask()` ........................................... 63
Table 2-31: Arguments for `GetRecordTask()` ............................................ 63
Table 2-32: Error Codes for `GetRecordTask()` ........................................... 64
Table 2-33: Arguments for `EnableRecordTask()` ....................................... 65
Table 2-34: Error Codes for `EnableRecordTask()` ....................................... 65
Table 2-35: Arguments for `DisableRecordTask()` ....................................... 66
Table 2-36: Error Codes for `DisableRecordTask()` ....................................... 66
Table 2-37: Arguments for `ResetRecordTask()` ........................................... 67
Table 2-38: Error Codes for `ResetRecordTask()` ........................................... 67
Table 2-39: Arguments for `GetRecordScheduleConflicts()` ....................... 68
Table 2-40: Error Codes for `GetRecordScheduleConflicts()` ....................... 69
Table 2-41: Arguments for `GetRecordTaskConflicts()` ................................ 69
Table 2-42: Error Codes for `GetRecordTaskConflicts()` ................................ 70
Table 2-43: Common Error Codes .............................................................. 70
Table 2-44: `recordTask` State Timeline ..................................................... 81
Table 2-45: Example 1: Fewer `recordSchedule` instances than the Number of Supported Priority Levels ................................................................. 82
Table 2-46: Example 2: More `recordSchedule` instances than the Number of Supported Priority Levels ................................................................. 83
Table 2-47: Existing `recordSchedule` Priorities .......................................... 84
Table 2-48: `desiredPriority` Property Set to “RS-C” ..................................... 84
Table 2-49: `desiredPriority` Property Set to “HIGHEST”, “L1_HI”, or “RS-A” ... 84
Table 2-50: `desiredPriority` Property Set to “LOWEST”, “L3_LOW”, or “RS-B” . . 85
Table 2-51: `desiredPriority` Property Set to “RS-C” ..................................... 85
Table B-1: Base Properties Overview .......................................................... 133
Table B-2: allowedValueList for the `class` Property .................................... 134
Table B-3: Priority Properties ..................................................................... 135
Table B-4: allowedValueList for the `priority` Property ................................ 136
Table B-5: Primary allowedValueList for the `desiredPriority` Property ............ 137
Table B-6: Additional allowedValueList for the `desiredPriority` Property .......... 138
Table B-7: allowedValueList for the `desiredPriority@type` Property ............. 139
Table B-8: Output Control Properties .......................................................... 139
Table B-9: `desiredRecordQuality` Example ............................................... 142
Table B-10: allowedValueList for the `desiredRecordQuality` Property .......... 143
Table B-11: allowedValueList for the `scheduledRecordQuality@type` Property . 143
Table B-12: Content Identification Related Properties .................................. 144
Table B-13: allowedValueList for the `scheduledChannelID@type` Property .... 146
<table>
<thead>
<tr>
<th>Table B-14:</th>
<th>Matching Content Criteria Properties .............................................................................</th>
<th>148</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table B-15:</td>
<td>allowedValueList for the <code>matchingName@type</code> Property ........................................</td>
<td>149</td>
</tr>
<tr>
<td>Table B-16:</td>
<td>allowedValueList for the <code>matchingID@type</code> Property ...........................................</td>
<td>150</td>
</tr>
<tr>
<td>Table B-17:</td>
<td>Matching Qualifying Criteria Properties ...................................................................</td>
<td>151</td>
</tr>
<tr>
<td>Table B-18:</td>
<td>allowedValueList for the <code>matchingRatingLimit</code> Property Using the MPAA Rating System (breakingRatingLimit@type = &quot;MPAA.ORG&quot;)</td>
<td>153</td>
</tr>
<tr>
<td>Table B-19:</td>
<td>allowedValueList for the <code>matchingRatingLimit</code> Property Using the RIAA Rating System (matchingRatingLimit@type = &quot;RIAA.ORG&quot;)</td>
<td>154</td>
</tr>
<tr>
<td>Table B-20:</td>
<td>allowedValueList for the <code>matchingRatingLimit</code> Property Using the ESRB Rating System (matchingRatingLimit@type = &quot;ESRB.ORG&quot;)</td>
<td>154</td>
</tr>
<tr>
<td>Table B-21:</td>
<td>allowedValueList for the <code>matchingRatingLimit</code> Property Using the TVGUIDELINES Rating System (matchingRatingLimit@type = &quot;TVGUIDELINES.ORG&quot;)</td>
<td>154</td>
</tr>
<tr>
<td>Table B-22:</td>
<td>allowedValueList for the <code>matchingRatingLimit@type</code> Property ....................................</td>
<td>155</td>
</tr>
<tr>
<td>Table B-23:</td>
<td>allowedValueList for the <code>matchingEpisodeType</code> Property ..........................................</td>
<td>156</td>
</tr>
<tr>
<td>Table B-24:</td>
<td>Content Control Properties .............................................................................................</td>
<td>156</td>
</tr>
<tr>
<td>Table B-25:</td>
<td>allowedValueList for the <code>durationLimit@effect</code> Property ...........................................</td>
<td>159</td>
</tr>
<tr>
<td>Table B-26:</td>
<td>Storage Related Properties ..................................................................................................</td>
<td>160</td>
</tr>
<tr>
<td>Table B-27:</td>
<td>Schedule State Properties ..................................................................................................</td>
<td>161</td>
</tr>
<tr>
<td>Table B-28:</td>
<td>allowedValueList for the <code>scheduleState</code> Property ......................................................</td>
<td>162</td>
</tr>
<tr>
<td>Table B-29:</td>
<td>allowedValueList for the <code>scheduleState@currentError</code> Property ....................................</td>
<td>162</td>
</tr>
<tr>
<td>Table B-30:</td>
<td>Statistics Properties .........................................................................................................</td>
<td>163</td>
</tr>
<tr>
<td>Table B-31:</td>
<td>Task General Properties .....................................................................................................</td>
<td>164</td>
</tr>
<tr>
<td>Table B-32:</td>
<td>Task Content Identification Properties ...............................................................................</td>
<td>165</td>
</tr>
<tr>
<td>Table B-33:</td>
<td><code>recordQuality</code> Example ...................................................................................................</td>
<td>169</td>
</tr>
<tr>
<td>Table B-34:</td>
<td>allowedValueList for the <code>recordQuality</code> Property .......................................................</td>
<td>170</td>
</tr>
<tr>
<td>Table B-35:</td>
<td>Task Matched Content Criteria Properties .........................................................................</td>
<td>170</td>
</tr>
<tr>
<td>Table B-36:</td>
<td>Task Matched Qualifying Criteria Properties ....................................................................</td>
<td>171</td>
</tr>
<tr>
<td>Table B-37:</td>
<td>Task Matched Control Properties .......................................................................................</td>
<td>172</td>
</tr>
<tr>
<td>Table B-38:</td>
<td>State Related Properties ....................................................................................................</td>
<td>174</td>
</tr>
<tr>
<td>Table B-39:</td>
<td>allowedValueList for the <code>taskState</code> Property ..................................................................</td>
<td>175</td>
</tr>
<tr>
<td>Table B-40:</td>
<td>allowedValueList for the <code>taskState</code> Property ..................................................................</td>
<td>176</td>
</tr>
<tr>
<td>Table B-41:</td>
<td>allowedValueList for the <code>taskState@phase</code> Property .....................................................</td>
<td>177</td>
</tr>
<tr>
<td>Table B-42:</td>
<td>allowedValueList for the <code>taskState@xxx</code> Properties .......................................................</td>
<td>180</td>
</tr>
<tr>
<td>Table C-1:</td>
<td>Class Properties Overview for <code>recordScheduleParts</code> usage ...........................................</td>
<td>191</td>
</tr>
<tr>
<td>Table C-2:</td>
<td>Class Properties Overview for <code>recordSchedule</code> usage ..................................................</td>
<td>195</td>
</tr>
<tr>
<td>Table C-3:</td>
<td>Class Properties Overview for <code>recordTask</code> usage .........................................................</td>
<td>199</td>
</tr>
<tr>
<td>Table C-4:</td>
<td><code>object</code> Base Class Properties .......................................................................................</td>
<td>203</td>
</tr>
<tr>
<td>Table C-5:</td>
<td><code>object.recordSchedule</code> Base Class Properties ..............................................................</td>
<td>204</td>
</tr>
</tbody>
</table>
Table C-6: `object.recordSchedule.direct` Class Properties ..........................................................205
Table C-7: `object.recordSchedule.direct.manual` Class Properties ........................................206
Table C-8: `object.recordSchedule.direct.cdsEPG` Class Properties ......................................207
Table C-9: `object.recordSchedule.direct.cdsNonEPG` Class Properties ..................................209
Table C-10: `object.recordSchedule.direct.programCode` Class Properties ..............................210
Table C-11: `object.recordSchedule.query` Class Properties ....................................................211
Table C-12: `object.recordSchedule.query.contentName` Class Properties ...............................212
Table C-13: `object.recordSchedule.query.contentID` Class Properties ....................................213
Table C-14: `object.recordTask` Base Class Properties ............................................................215
List of Figures

Figure 1: Creating a new recordSchedule ................................................................. 35
Figure 2: Capability check. ...................................................................................... 35
Figure 3: Browse recordSchedule .......................................................................... 36
Figure 4: Delete a recordSchedule ........................................................................ 36
Figure 5: A Full-Featured State Diagram ................................................................. 73
Figure 6: A Minimal-Implementation State Diagram ............................................. 78
Figure 7: Class hierarchy for the ScheduledRecording service.............................. 188
1 Overview and Scope

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

1.1 Introduction

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

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

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

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

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

This service template does not address:

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

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

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

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

  PROHIBITED – The definition or behavior is prohibited by this specification. Opposite of REQUIRED.

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

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

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

- Strings that are to be taken literally are enclosed in “double quotes”.

- Words that are emphasized are printed in italic.

- Keywords that are defined by the UPnP AV Working Committee are printed using the forum character style.

- Keywords that are defined by the UPnP Device Architecture specification are printed using the arch character style [DEVICE].

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

1.2.1 Data Types

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

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

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

1.2.2 Strings Embedded in Other Strings

Some string variables and arguments described in this document contain substrings that MUST be independently identifiable and extractable for other processing. This requires the definition of appropriate substring delimiters and an escaping mechanism so that these delimiters can also appear as ordinary
characters in the string and/or its independent substrings. This document uses embedded strings in two contexts – Comma Separated Value (CSV) lists (see Section 1.3.1, “Comma Separated Value (CSV) Lists”) and property values in search criteria strings. Escaping conventions use the backslash character, “\” (character code U+005C), as follows:

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

1.2.3 Extended Backus-Naur Form

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

1.2.3.1 Typographic conventions for EBNF

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>::=</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>alternative separator – separates sequences on the right that are independently allowed definitions for the non-terminal on the left.</td>
</tr>
<tr>
<td>*</td>
<td>null repetition – means the expression to its left MAY occur zero or more times.</td>
</tr>
<tr>
<td>+</td>
<td>non-null repetition – means the expression to its left MUST occur at least once and MAY occur more times.</td>
</tr>
<tr>
<td>[ ]</td>
<td>optional – the expression between the brackets is optional.</td>
</tr>
<tr>
<td>( )</td>
<td>grouping – groups the expressions between the parentheses.</td>
</tr>
<tr>
<td>-</td>
<td>character range – represents all characters between the left and right character operands inclusively.</td>
</tr>
</tbody>
</table>

1.3 Derived Data Types

This section defines a derived data type that is represented as a string data type with special syntax. This specification uses string data type definitions that originate from two different sources. The UPnP Device Architecture defined *string* data type is used to define state variable and action argument *string* data types.

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
The XML Schema namespace is used to define property xsd:string data types. The following definition applies to both string data types.

### 1.3.1 Comma Separated Value (CSV) Lists

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

- A list is represented as a `string` type (for state variables and action arguments) or xsd:string type (for properties).
- Commas separate values within a list.
- Integer values are represented in CSVs with the same syntax as the integer data type specified in [DEVICE] (that is: optional leading sign, optional leading zeroes, numeric US-ASCII).
- Boolean values are represented in state variable and action argument CSVs as either “0” for false or “1” for true. These values are a subset of the defined `boolean` data type values specified in [DEVICE]: 0, false, no, 1, true, yes.
- Boolean values are represented in property CSVs as either “0” for false or “1” for true. These values are a subset of the defined Boolean data type values specified in [XML SCHEMA-2]: 0, false, 1, true.
- Escaping conventions for the comma and backslash characters are defined in Section 1.2.2, “Strings Embedded in Other Strings”.
- White space before, after, or interior to any numeric data type is not allowed.
- White space before, after, or interior to any other data type is part of the value.

**Table 1-2: CSV Examples**

<table>
<thead>
<tr>
<th>Type refinement of string</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV (<code>string</code>) or CSV (xsd:string)</td>
<td>“+artist,-date”</td>
<td>List of 2 property sort criteria.</td>
</tr>
<tr>
<td>CSV (<code>int</code>) or CSV (xsd:integer)</td>
<td>“1,-5,006,0,+7”</td>
<td>List of 5 integers.</td>
</tr>
<tr>
<td>CSV (<code>boolean</code>) or CSV (xsd:Boolean)</td>
<td>“0,1,1,0”</td>
<td>List of 4 booleans</td>
</tr>
<tr>
<td>CSV (<code>string</code>) or CSV (xsd:string)</td>
<td>“Smith, Fred, Jones, Davey”</td>
<td>List of 2 names, “Smith, Fred” and “Jones, Davey”</td>
</tr>
</tbody>
</table>
### Type refinement of string

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV (i4, string, ui2) or CSV (xsd:int, xsd:string, xsd:unsignedShort)</td>
<td>“-29837, string with leading blanks,0”</td>
<td>Note that the second value is “string with leading blanks”</td>
</tr>
<tr>
<td>CSV (i4) or CSV (xsd:int)</td>
<td>“3, 4”</td>
<td>Illegal CSV. White space is not allowed as part of an integer value.</td>
</tr>
<tr>
<td>CSV (string) or CSV (xsd:string)</td>
<td>“,”</td>
<td>List of 3 empty string values</td>
</tr>
<tr>
<td>CSV (heterogeneous)</td>
<td>“Alice,Marketing,5,Sue,R&amp;D,21,Dave,Finance,7”</td>
<td>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.</td>
</tr>
</tbody>
</table>

### 1.4 Management of XML Namespaces in Standardized DCPs

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

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

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

For the same reason that namespace prefixes are convenient in XML documents, it is convenient in specification text to refer to namespaces using a namespace prefix. Therefore, this specification declares a “standard” prefix for all XML namespaces used herein. In addition, this specification expands the scope where these prefixes have meaning, beyond a single XML document, to all of its text, XML examples, and certain string-valued properties. This expansion of scope does not 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, MUST use
the predefined namespace prefixes when referring to CDS properties ("upnp:","dc:", etc).

All of the namespaces used in this specification are listed in the Tables “Namespace Definitions” and
“Schema-related Information”. For each such namespace, Table 1-3, “Namespace Definitions” gives a
brief description of it, its name (a URI) and its defined “standard” prefix name. Some namespaces included
in these tables are not directly used or referenced in this document. They are included for completeness to
accommodate those situations where this specification is used in conjunction with other UPnP
specifications to construct a complete system of devices and services. For example, since the Scheduled
Recording Service depends on and refers to the Content Directory Service, the predefined “srs:”
namespace prefix is included. The individual specifications in such collections all use the same standard
prefix. The standard prefixes are also used in Table 1-4, “Schema-related Information”, to cross-reference
additional namespace information. This second table includes each namespace’s valid XML document root
element(s) (if any), its schema file name, versioning information (to be discussed in more detail below),
and a link to the entry in Section 1.6, “References” for its associated schema.

The normative definitions for these namespaces are the documents referenced in Table 1-3. The schemas
are designed to support these definitions for both human understanding and as test tools. However,
limitations of the XML Schema language itself make it difficult for the UPnP-defined schemas to
accurately represent all details of the namespace definitions. As a result, the schemas will validate many
XML documents that are not valid according to the specifications.

The Working Committee expects to continue refining these schemas after specification release to reduce
the number of documents that are validated by the schemas while violating the specifications, but the
schemas will still be informative, supporting documents. Some schemas might become normative in future
versions of the specifications.

<table>
<thead>
<tr>
<th>Standard Namespace Prefix</th>
<th>Namespace Name</th>
<th>Namespace Description</th>
<th>Normative Definition Document Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>av</td>
<td>urn:schemas-upnp-org:av:av</td>
<td>Common data types for use in AV schemas</td>
<td>[AV-XSD]</td>
</tr>
<tr>
<td>avs</td>
<td>urn:schemas-upnp-org:av:avs</td>
<td>Common structures for use in AV schemas</td>
<td>[AVS-XSD]</td>
</tr>
<tr>
<td>avdt</td>
<td>urn:schemas-upnp-org:av:avdt</td>
<td>Datastructure Template</td>
<td>[AVDT]</td>
</tr>
<tr>
<td>avt-event</td>
<td>urn:schemas-upnp-org:metadata-1-0/AVT/</td>
<td>Evented LastChange state variable for AVTransport</td>
<td>[AVT]</td>
</tr>
<tr>
<td>cds-event</td>
<td>urn:schemas-upnp-org:av:cds-event</td>
<td>Evented LastChange state variable for ContentDirectory</td>
<td>[CDS]</td>
</tr>
<tr>
<td>didl-lite</td>
<td>urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/</td>
<td>Structure and metadata for ContentDirectory</td>
<td>[CDS]</td>
</tr>
<tr>
<td>rcs-event</td>
<td>urn:schemas-upnp-org:metadata-1-0/RCS/</td>
<td>Evented LastChange state variable for RenderingControl</td>
<td>[RCS]</td>
</tr>
<tr>
<td>srs</td>
<td>urn:schemas-upnp-org:av:srs</td>
<td>Metadata and structure for ScheduledRecording</td>
<td>[SRS]</td>
</tr>
<tr>
<td>srs-event</td>
<td>urn:schemas-upnp-org:av:srs-event</td>
<td>Evented LastChange state variable for ScheduledRecording</td>
<td>[SRS]</td>
</tr>
<tr>
<td>upnp</td>
<td>urn:schemas-upnp-org:metadata-1-0/upnp/</td>
<td>Metadata for ContentDirectory</td>
<td>[CDS]</td>
</tr>
<tr>
<td>Standard Name-space Prefix</td>
<td>Namespace Name</td>
<td>Namespace Description</td>
<td>Normative Definition Document Reference</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------</td>
<td>-----------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td><strong>Externally defined namespaces</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dc</td>
<td><a href="http://purl.org/dc/elements/1.1/">http://purl.org/dc/elements/1.1/</a></td>
<td>Dublin Core</td>
<td>[DC-TERMS]</td>
</tr>
<tr>
<td>xsd</td>
<td><a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a></td>
<td>XML Schema Language 1.0</td>
<td>[XML SCHEMA-1] [XML SCHEMA-2]</td>
</tr>
<tr>
<td>xsi</td>
<td><a href="http://www.w3.org/2001/XMLSchema-instance">http://www.w3.org/2001/XMLSchema-instance</a></td>
<td>XML Schema Instance Document schema</td>
<td>Sections 2.6 &amp; 3.2.7 of [XML SCHEMA-1]</td>
</tr>
</tbody>
</table>

**Table 1-4: Schema-related Information**

<table>
<thead>
<tr>
<th>Standard Name-space Prefix</th>
<th>Relative URI and File Name</th>
<th>Valid Root Element(s)</th>
<th>Schema Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AV Working Committee Defined Namespaces</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>av</td>
<td>av-vn-yyyyymmddd.xsd</td>
<td>n/a</td>
<td>[AV-XSD]</td>
</tr>
<tr>
<td></td>
<td>av-vn.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>av.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>avs</td>
<td>avs-vn-yyyyymmddd.xsd</td>
<td>&lt;Capabilities&gt;</td>
<td>[AVS-XSD]</td>
</tr>
<tr>
<td></td>
<td>avs-vn.xsd</td>
<td>&lt;Features&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>avs.xsd</td>
<td>&lt;stateVariableValuePairs&gt;</td>
<td></td>
</tr>
<tr>
<td>avdt</td>
<td>avdt-vn-yyyyymmddd.xsd</td>
<td>&lt;AVDT&gt;</td>
<td>[AVDT]</td>
</tr>
<tr>
<td></td>
<td>avdt-vn.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>avdt.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>avt-event</td>
<td>avt-event-vn-yyyyymmddd.xsd</td>
<td>&lt;Event&gt;</td>
<td>[AVT-EVENT-XSD]</td>
</tr>
<tr>
<td></td>
<td>avt-event-vn.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>avt-event.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cds-event</td>
<td>cds-event-vn-yyyyymmddd.xsd</td>
<td>&lt;StateEvent&gt;</td>
<td>[CDS-EVENT-XSD]</td>
</tr>
<tr>
<td></td>
<td>cds-event-vn.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cds-event.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>didl-lite</td>
<td>didl-lite-vn-yyyyymmddd.xsd</td>
<td>&lt;DIDL-Lite&gt;</td>
<td>[DIDL-LITE-XSD]</td>
</tr>
<tr>
<td></td>
<td>didl-lite-vn.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>didl-lite.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rcs-event</td>
<td>rcs-event-vn-yyyyymmddd.xsd</td>
<td>&lt;Event&gt;</td>
<td>[RCS-EVENT-XSD]</td>
</tr>
<tr>
<td></td>
<td>rcs-event-vn.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rcs-event.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>srs</td>
<td>srs-vn-yyyyymmddd.xsd</td>
<td>&lt;srs&gt;</td>
<td>[SRS-XSD]</td>
</tr>
<tr>
<td></td>
<td>srs-vn.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>srs.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>srs-event</td>
<td>srs-event-vn-yyyyymmddd.xsd</td>
<td>&lt;StateEvent&gt;</td>
<td>[SRS-EVENT-XSD]</td>
</tr>
<tr>
<td></td>
<td>srs-event-vn.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>srs-event.xsd</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Standard Namespace Prefix

<table>
<thead>
<tr>
<th>Standard Namespace Prefix</th>
<th>Relative URI and File Name</th>
<th>Valid Root Element(s)</th>
<th>Schema Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>upnp</td>
<td>upnp-vn-yyyyymmdd.xsd</td>
<td>n/a</td>
<td>[UPNP-XSD]</td>
</tr>
<tr>
<td></td>
<td>upnp-vn.xsd</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>upnp.xsd</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Externally Defined Namespaces

<table>
<thead>
<tr>
<th>Namespace</th>
<th>Absolute URL</th>
<th>Schema Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc</td>
<td>Absolute URL: <a href="http://dublincore.org/schemas/xmls/simpledc20021212.xsd">http://dublincore.org/schemas/xmls/simpledc20021212.xsd</a></td>
<td>[DC-XSD]</td>
</tr>
<tr>
<td>xsd</td>
<td>n/a</td>
<td>&lt;schema&gt;</td>
</tr>
<tr>
<td>xsi</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>xml</td>
<td>n/a</td>
<td></td>
</tr>
</tbody>
</table>

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

### 1.4.1 Namespace Prefix Requirements

There are many occurrences in this specification of string data types that contain XML names (property names). These XML names in strings will not be processed under namespace-aware conditions. Therefore, all occurrences in instance documents of XML names in strings MUST use the standard namespace prefixes as declared in Table 1-3. In order to properly process the XML documents described herein, control points and devices MUST use namespace-aware XML processors [XML-NMSP] for both reading and writing. As allowed by [XML-NMSP], the namespace prefixes used in an instance document are at the sole discretion of the document creator. Therefore, the declared prefix for a namespace in a document MAY be different from the standard prefix. All devices MUST be able to correctly process any valid XML instance document, even when it uses a non-standard prefix for ordinary XML names. However, it is strongly RECOMMENDED that all devices use these standard prefixes for all instance documents to avoid confusion on the part of both human and machine readers. These standard prefixes are used in all descriptive text and all XML examples in this and related UPnP specifications. Also, each individual specification may assume a default namespace for its descriptive text. In that case, names from that namespace may appear with no prefix.

The assumed default namespace, if any, for each UPnP AV specification is given in Table 1-5, “Default Namespaces for the AV Specifications”.

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

### Table 1-5: Default Namespaces for the AV Specifications

<table>
<thead>
<tr>
<th>AV Specification Name</th>
<th>Default Namespace Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVTransport</td>
<td>avt-event</td>
</tr>
<tr>
<td>ConnectionManager</td>
<td>n/a</td>
</tr>
<tr>
<td>ContentDirectory</td>
<td>didl-lite</td>
</tr>
<tr>
<td>MediaRenderer</td>
<td>n/a</td>
</tr>
<tr>
<td>MediaServer</td>
<td>n/a</td>
</tr>
<tr>
<td>RenderingControl</td>
<td>rcs-event</td>
</tr>
<tr>
<td>ScheduledRecording</td>
<td>srs</td>
</tr>
</tbody>
</table>

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
1.4.2 Namespace Names, Namespace Versioning and Schema Versioning

The UPnP AV service specifications define several data structures (such as state variables and action arguments) whose format is an XML instance document that must comply with one or more specific XML namespaces. Each namespace is uniquely identified by an assigned namespace name. The namespaces that are defined by the AV Working Committee MUST be named by a URN. See Table 1-3, “Namespace Definitions” for a current list of namespace names. Additionally, each namespace corresponds to an XML schema document that provides a machine-readable representation of the associated namespace to enable automated validation of the XML (state variable or action parameter) instance documents.

Within an XML schema and XML instance document, the name of each corresponding namespace appears as the value of an `xmlns` attribute within the root element. Each `xmlns` attribute also includes a namespace prefix that is associated with that namespace in order to disambiguate (a.k.a. qualify) element and attribute names that are defined within different namespaces. The schemas that correspond to the listed namespaces are identified by URI values that are listed in the `schemaLocation` attribute also within the root element. (See Section 1.4.3, “Namespace Usage Examples”)

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

Each time a new namespace version is created, a new XML schema document (.xsd) is created and published so that the new namespace definition is represented in a machine-readable form. Since a XML schema document is just a representation of a namespace definition, translation errors can occur. Therefore, it is sometime necessary to re-release a published schema in order to correct typos or other namespace representation errors. In order to easily identify the potential multiplicity of schema releases for the same namespace, the URI of each released schema MUST conform to the following format (called Form 1):

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

where

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

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

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

In addition to the dated schema URIs that are associated with each namespace, each namespace also has a set of undated schema URIs. These undated schema URIs have two distinct formats with slightly different meanings:
Form 2: “http://www.upnp.org/schemas/av/” schema-root-name “-v” ver
where ver is described above.

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

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

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

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

- All instance documents, whether generated by a service or a control point, MUST use Form 3.
- All UPnP AV published schemas that reference other UPnP AV schemas MUST also use Form 3.

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

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

ver “-” yyyyymmdd where ver and yyyyymmdd are described above.

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

### 1.4.3 Namespace Usage Examples

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

**Example 1:**

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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
```
1.5 Vendor-defined Extensions

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

1.5.1 Vendor-defined Action Names

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

1.5.2 Vendor-defined State Variable Names

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

1.5.3 Vendor-defined XML Elements and attributes

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

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
1.5.4 Vendor-defined Property Names

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

1.6 References

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


Available at: http://www.faqs.org/rfc/rfc2119.html.

Available at: http://www.ietf.org/rfc/rfc2396.txt.

Available at: http://www.ietf.org/rfc/rfc3339.txt.

Available at: http://www.ietf.org/rfc/rfc1889.txt.

Available at: http://www.ietf.org/rfc/rfc2326.txt.

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

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

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

Available at: http://www.unicode.org/reports/tr15/tr15-25.html.

Available at: http://www.unicode.org/reports/tr10/tr10-14.html.

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

Available at: http://www.unicode.org/reports/tr10/tr10-14.html.

Available at: http://www.unicode.org/reports/tr35/tr35-5.html.

Available at: http://www.w3.org/TR/2004/REC-xml-20040204.


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


2 Service Modeling Definitions

2.1 ServiceType

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

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

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

2.2 Terms and Abbreviations

2.2.1 Abbreviations

Table 2-1: Abbreviations

<table>
<thead>
<tr>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDS</td>
<td>ContentDirectory Service</td>
</tr>
<tr>
<td>EPG</td>
<td>Electronic Program Guide</td>
</tr>
<tr>
<td>SRS</td>
<td>ScheduledRecording Service</td>
</tr>
</tbody>
</table>

2.2.2 Terms

2.2.2.1 CDS object

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

2.2.2.2 User Channel

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

2.2.2.3 Channel Group

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

2.2.2.4 Channel Line-up

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

2.2.2.5 object

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

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

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

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

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

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

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

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

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

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

2.2.2.13 Property-set Data Types
Certain ScheduledRecording service actions use property-set arguments that contain information about a set of properties, typically expressed in the form of an srs XML Document (for example, the Elements argument of the CreateRecordSchedule() action). The set of properties that can exist in a property-set argument is implementation dependent. Indeed, the set of optional properties that a particular ScheduledRecording service chooses to implement is vendor dependent.
This specification currently defines three different *property-set* data types:

- `A_ARG_TYPE_RecordSchedule`
- `A_ARG_TYPE_RecordTask`
- `A_ARG_TYPE_RecordScheduleParts`

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

### 2.2.2.14 Property

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

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

**Examples:**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>XML Representation (srs declared as default namespace)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>title</code></td>
<td><code>&lt;title&gt;…&lt;/title&gt;</code></td>
</tr>
<tr>
<td><code>taskProgramCode</code></td>
<td><code>&lt;taskProgramCode&gt;…&lt;/taskProgramCode&gt;</code></td>
</tr>
<tr>
<td><code>taskProgramCode@type</code></td>
<td><code>&lt;taskProgramCode type=&quot;...&quot;&gt;…&lt;/taskProgramCode&gt;</code></td>
</tr>
<tr>
<td><code>@id</code></td>
<td><code>&lt;item id=&quot;...&quot;&gt;…&lt;/item&gt;</code></td>
</tr>
</tbody>
</table>

### 2.2.2.15 Member Property

A property is a member of a particular class when the property is declared to be either REQUIRED or OPTIONAL for that class.
2.2.2.16 Supported Member Property
A supported member property is a member property that is supported by a particular ScheduledRecording service implementation, according to the information returned by the GetPropertyList() action.

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

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

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

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

Therefore, the string containing the srs XML Document will have one of the following two forms:
“<srs ...>...</srs>”

or
“<?xml ...?>
<srs ...>...</srs>”

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

For example, the phrase “srs XML Fragment” refers to a sequence of XML elements that are defined in the SRS Schema as defined in [SRS-XSD]:
“<item id="..." ...>...</item>”

or
“<recordDestination mediaType="..." preference="...”>
...
</recordDestination>”

or
“<title>...</title>
<class>...</class>
<...>...</>
...
<...>...</>”

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
2.2.2.21 actualScheduledStartDateTime
The actual scheduled start date&time of a program item is defined as:

\[
\text{actualScheduledStartDateTime} = \text{scheduledStartDateTime} + \text{scheduledStartDateTimeAdjust}
\]

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

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

\[
\text{actualStartDateTime} = \text{actualScheduledStartDateTime} + \text{any device-specific record startup latency}
\]

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

\[
\text{actualScheduledEndDateTime} = \text{scheduledStartDateTime} + \text{scheduledDuration} + \text{scheduledDurationAdjust}
\]

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

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

\[
\text{actualEndDateTime} = \text{actualScheduledEndDateTime} + \text{any device-specific record teardown latency}
\]

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

\[
\text{actualScheduledDuration} = \text{actualScheduledEndDateTime} - \text{actualScheduledStartDateTime}
\]

\[
= \text{scheduledDuration} + \text{scheduledDurationAdjust} - \text{scheduledStartDateTimeAdjust}
\]

where \( \text{scheduledDuration} \) is the scheduled broadcast duration of the program item, \( \text{scheduledDurationAdjust} \) is a user-supplied adjustment to that duration, and \( \text{scheduledStartDateTimeAdjust} \) is a user-supplied adjustment to the scheduled start date&time.

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

Simple example: one of the most familiar examples is case-insensitive sorting on the ASCII subset of Unicode. In a binary ASCII sort, all lower case letters sort after the upper case “Z” because “Z” has a character code of 0x5A, and all lower case character codes are greater than or equal to 0x61.
More complex example: the “ö” character in German sorts between “n” and “p” characters whereas in Swedish, it sorts after “z”.

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

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

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

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

upper(c) ≤ lower(c)
upper(c) < lower(c)
upper(c) = lower(c)

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

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

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

2.2.2.32 Sequenced Sort

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

2.2.2.33 Sequenced Lexical Sort

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

2.2.2.34 Sequenced Numeric Sort

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

2.2.2.35 Lexical Numeric Sort

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

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

2.2.2.36 type Relationship Sort

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

2.3 ScheduledRecording Service Architecture

2.3.1 recordSchedule

A ScheduledRecording service implementation has a single, flat list of `recordSchedule` instances. A `recordSchedule` represents the user-level recording instructions to the ScheduledRecording service. These user-level instructions have various levels of complexity. For example, a simple instruction may state: “record channel 15 at 4PM on March 19, 2004,” while a more complex instruction may state: “record all episodes of the DIY Home Improvement Show on any channel that has the show for the next month.” The behavior of a `recordSchedule` is described by one or more properties, and these properties can be manipulated through several actions.
As shown in Figure 1, when a control point requests a new scheduled recording to the ScheduledRecording service via the `CreateRecordSchedule()` action, the control point sets a number of properties and passes them to the ScheduledRecording service to express user-desired instructions to the scheduled recording. Then, as a response to the `CreateRecordSchedule()` action, the ScheduledRecording service creates a `recordSchedule`, assigns a unique ID to the `recordSchedule` and returns the `recordSchedule` with the complete set of initial property settings. The ScheduledRecording service MUST add OPTIONAL properties to the `recordSchedule` when a control point did not specify them. Additionally, the ScheduledRecording service MAY add some informative properties.

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

![Figure 1: Creating a new recordSchedule](image)

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

![Figure 2: Capability check.](image)

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.

![Diagram](image1.png)

Figure 4: Delete a recordSchedule

### 2.3.2 recordTask

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

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

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

### 2.4 State Variables

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

---

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
information via the return arguments of the actions defined below. The majority of state variables defined below exist simply to provide data type information for the arguments of the various actions of this service.

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

### 2.4.1 State Variable Overview

Table 2-2: State Variables

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>R/O1</th>
<th>Data Type</th>
<th>Allowed Value</th>
<th>Default Value</th>
<th>Eng. Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>SortCapabilities</td>
<td>R</td>
<td>string</td>
<td>CSV (string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SortLevelCapability</td>
<td>R</td>
<td>ui4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StateUpdateID</td>
<td>R</td>
<td>ui4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LastChange</td>
<td>R</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_PropertyList</td>
<td>R</td>
<td>string</td>
<td>CSV (string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_DataTypeID</td>
<td>R</td>
<td>string</td>
<td>See Table 2-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_ObjectID</td>
<td>R</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_ObjectIDList</td>
<td>O</td>
<td>string</td>
<td>CSV (string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_PropertyInfo</td>
<td>R</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_Index</td>
<td>R</td>
<td>ui4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_Count</td>
<td>R</td>
<td>ui4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_SortCriteria</td>
<td>R</td>
<td>string</td>
<td>CSV (string)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_RecordSchedule</td>
<td>R</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_RecordTask</td>
<td>R</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_RecordScheduleParts</td>
<td>R</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O1</th>
</tr>
</thead>
<tbody>
<tr>
<td>“A_ARG_TYPE_RecordSchedule”</td>
<td>R</td>
</tr>
<tr>
<td>“A_ARG_TYPE_RecordTask”</td>
<td>R</td>
</tr>
<tr>
<td>“A_ARG_TYPE_RecordScheduleParts”</td>
<td>R</td>
</tr>
<tr>
<td>Vendor-defined</td>
<td>X</td>
</tr>
</tbody>
</table>
2.4.2 **SortCapabilities**

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

2.4.3 **SortLevelCapability**

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

2.4.4 **StateUpdateID**

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

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

For example, consider the case where a control point invokes **CreateRecordSchedule** to create a new **recordSchedule** that also immediately spawns exactly one **recordTask**. Assume that **StateUpdateID** is 10 when the control point invokes the action and that for a short time period around this invocation, no other activity occurs that affects the value of **StateUpdateID**. During this time period, exactly one of the following MUST be true as seen by all external observations (including the returned values from this **CreateRecordSchedule** invocation):

- **StateUpdateID** is 10; and the new **recordSchedule** has not been created; and the new **recordTask** has not been created.
- **StateUpdateID** is 11; and the **recordSchedule** has been created; and the new **recordTask** has not been created; and the **recordSchedule**’s value of **currentRecordTaskCount** is 0, indicating that no **recordTask** has been created.
- **StateUpdateID** is 12; and the **recordSchedule** has been created; and the new **recordTask** has been created; and the **recordSchedule**’s value of **currentRecordTaskCount** is 1, indicating that the child **recordTask** has been created.

ScheduledRecording service implementations SHOULD maintain the same value for **StateUpdateID** through power cycles and any other disappearance/reappearance of the service on the network. Control points can use a change in the value of this variable to determine if there has been a change in the ScheduledRecording service.
The value of the `StateUpdateID` state variable, returned within events and returned as an output argument of certain actions should be monitored very closely by control points. Indeed, whenever an action returns with a `StateUpdateID` value in its `UpdateID` argument that is less than the `StateUpdateID` value received in the `updateID` attribute from the most recent `LastChange` event, the information returned by that action is potentially stale. A control point may want to refresh that information for instance by invoking the appropriate `Browsexxx()` or `Getxxx()` action. It is safe to use the information as long as the `StateUpdateID` value returned in the `UpdateID` argument of the action is greater than or equal to the `StateUpdateID` value received in the `updateID` attribute from the most recent `LastChange` event.

### 2.4.5 LastChange

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

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

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordScheduleCreated</td>
<td>A new <code>recordSchedule</code> is created.</td>
</tr>
<tr>
<td>RecordScheduleModified</td>
<td>One or more properties of a <code>recordSchedule</code> are modified (added, deleted or values are changed).</td>
</tr>
<tr>
<td>RecordScheduleDeleted</td>
<td>A <code>recordSchedule</code> is deleted.</td>
</tr>
<tr>
<td>RecordTaskCreated</td>
<td>A new <code>recordTask</code> is created.</td>
</tr>
<tr>
<td>RecordTaskModified</td>
<td>One or more properties of a <code>recordTask</code> are modified (added, deleted or values are changed).</td>
</tr>
<tr>
<td>RecordTaskDeleted</td>
<td>A <code>recordTask</code> is deleted.</td>
</tr>
<tr>
<td>Vendor-defined</td>
<td>See Section 1.5, “Vendor-defined Extensions”.</td>
</tr>
</tbody>
</table>

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

**Example** (before XML escaping)

```xml
<?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=""
```
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"/>
  </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"/>
  </StateEvent>
GENA behavior: Send initial Notify with the \textit{LastChange} value above.

3: \textbf{ScheduledRecording service activity} = a \textit{recordTask} with \texttt{@id} = “t001-000” is created. Its associated \textit{recordSchedule} with \texttt{@id} = “s001” is modified by the ScheduledRecording service at the same time because its \texttt{currentRecordTaskCount} property is updated to reflect the existence of the new \textit{recordTask}.

\texttt{StateUpdateID} = 3

\texttt{LastChange} (before XML escaping):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
xmlns="urn:schemas-upnp-org:av:srs-event"
xmns: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 \texttt{LastChange} value above.

4: \textbf{ScheduledRecording service activity} = a \textit{recordTask} with \texttt{@id} = “t001-001” is created. Its associated \textit{recordSchedule} with \texttt{@id} = “s001” is modified by the ScheduledRecording service at the same time because its \texttt{currentRecordTaskCount} property is updated to reflect the existence of the new \textit{recordTask}. Within the same moderation period, a \textit{recordTask} with \texttt{@id} = “t001-002” is also created. Its associated \textit{recordSchedule} with \texttt{@id} = “s001” is modified by the ScheduledRecording service at the same time because its \texttt{currentRecordTaskCount} property is updated to reflect the existence of the new \textit{recordTask}.

\texttt{StateUpdateID} = 7

\texttt{LastChange} (before XML escaping):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StateEvent
xmlns="urn:schemas-upnp-org:av:srs-event"
xmns: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 \texttt{LastChange} value above.

2.4.6 \textbf{A ARG TYPE PropertyList}

This state variable is introduced to provide type information for various action arguments that contain a CSV list of property names. Namespace prefixes MUST be included with all property names (see Section
1.4, “Management of XML Namespaces”). The exact semantics of these property names depend on the associated action.

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

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

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

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

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

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

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

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

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

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

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

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

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

2.4.15 **A_ARG_TYPE_RecordTask**

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

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

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

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

2.4.16 **A_ARG_TYPE_RecordScheduleParts**

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

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

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

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

2.5 Eventing and Moderation

Table 2-5: Eventing and Moderation

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Evented</th>
<th>Moderated Event</th>
<th>Max Event Rate</th>
<th>Logical Combination</th>
<th>Min Delta per Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>SortCapabilities</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SortLevelCapability</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StateUpdateID</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LastChange</td>
<td>YES</td>
<td>YES</td>
<td>0.2 seconds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_PropertyList</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_DataTypeID</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_ObjectID</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_ObjectIDList</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_PropertyInfo</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_Index</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_Count</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_SortCriteria</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_RecordSchedule</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_RecordTask</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A_ARG_TYPE_RecordScheduleParts</td>
<td>NO</td>
<td>NO</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 Determined by N, where Rate = (Event)/(N secs).
2 (N) * (allowedValueRange Step).

2.6 Actions

Table 2-6: Actions

<table>
<thead>
<tr>
<th>Name</th>
<th>R/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetSortCapabilities()</td>
<td>R</td>
</tr>
</tbody>
</table>
### 2.6.1 GetSortCapabilities()

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

---

<table>
<thead>
<tr>
<th>Name</th>
<th>R/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPropertyList()</td>
<td>R</td>
</tr>
<tr>
<td>GetAllowedValues()</td>
<td>R</td>
</tr>
<tr>
<td>GetStateUpdateID()</td>
<td>R</td>
</tr>
<tr>
<td>BrowseRecordSchedules()</td>
<td>R</td>
</tr>
<tr>
<td>BrowseRecordTasks()</td>
<td>R</td>
</tr>
<tr>
<td>CreateRecordSchedule()</td>
<td>R</td>
</tr>
<tr>
<td>DeleteRecordSchedule()</td>
<td>R</td>
</tr>
<tr>
<td>GetRecordSchedule()</td>
<td>R</td>
</tr>
<tr>
<td>EnableRecordSchedule()</td>
<td>O²</td>
</tr>
<tr>
<td>DisableRecordSchedule()</td>
<td>O²</td>
</tr>
<tr>
<td>DeleteRecordTask()</td>
<td>O</td>
</tr>
<tr>
<td>GetRecordTask()</td>
<td>R</td>
</tr>
<tr>
<td>EnableRecordTask()</td>
<td>O³</td>
</tr>
<tr>
<td>DisableRecordTask()</td>
<td>O³</td>
</tr>
<tr>
<td>ResetRecordTask()</td>
<td>O³</td>
</tr>
<tr>
<td>GetRecordScheduleConflicts()</td>
<td>O⁴</td>
</tr>
<tr>
<td>GetRecordTaskConflicts()</td>
<td>O⁵</td>
</tr>
</tbody>
</table>

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

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

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

4 CONDITIONALLY REQUIRED. See Section 2.6.17, “GetRecordScheduleConflicts()” to determine when this action MUST be implemented.

5 CONDITIONALLY REQUIRED. See Section 2.6.18, “GetRecordTaskConflicts()” to determine when this action MUST be implemented.
2.6.1.1 Arguments

Table 2-7: Arguments for GetSortCapabilities()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>SortCaps</td>
<td>OUT</td>
<td>SortCapabilities</td>
</tr>
<tr>
<td>SortLevelCap</td>
<td>OUT</td>
<td>SortLevelCapability</td>
</tr>
</tbody>
</table>

2.6.1.1.1 SortCaps

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

2.6.1.1.2 SortLevelCap

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

2.6.1.2 Dependency on State

None.

2.6.1.3 Effect on State

None.

2.6.1.4 Errors

Table 2-8: Error Codes for GetSortCapabilities()

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
</tbody>
</table>

2.6.2 GetPropertyList()

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

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

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
The set of allowed values for srs properties and vendor-defined properties (when used for the specified property-set data type) can be obtained via the `GetAllowedValues()` action. The set of allowed values for imported properties cannot be retrieved by the `GetAllowedValues()` action.

### 2.6.2.1 Arguments

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DataTypeID</code></td>
<td>IN</td>
<td><code>A_ARG_TYPE_DataTypeID</code></td>
</tr>
<tr>
<td><code>PropertyList</code></td>
<td>OUT</td>
<td><code>A_ARG_TYPE_PropertyList</code></td>
</tr>
</tbody>
</table>

#### 2.6.2.1.1 `DataTypeID`

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

#### 2.6.2.1.2 `PropertyList`

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

### 2.6.2.2 Dependency on State

None.

### 2.6.2.3 Effect on State

None.

### 2.6.2.4 Errors

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

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>711</td>
<td>Invalid <code>DataTypeID</code></td>
<td>An invalid value has been specified in the <code>DataTypeID</code> input argument.</td>
</tr>
</tbody>
</table>

### 2.6.3 `GetAllowedValues()`

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

### 2.6.3.1 Arguments

**Table 2-11: Arguments for `getAllowedValues()`**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DataTypeID</code></td>
<td>IN</td>
<td>A_ARG_TYPE_DataTypeID</td>
</tr>
<tr>
<td><code>Filter</code></td>
<td>IN</td>
<td>A_ARG_TYPE_PropertyList</td>
</tr>
<tr>
<td><code>PropertyInfo</code></td>
<td>OUT</td>
<td>A_ARG_TYPE_PropertyInfo</td>
</tr>
</tbody>
</table>

#### 2.6.3.1.1 `DataTypeID`  
See Section 2.6.2.1.1, “`DataTypeID`”.

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

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

Examples of valid `Filter` argument values include:

- “srs:@id,srs:priority@orderedValue”
- “srs:title,srs:class”
- “*:*”
- “srs:*”

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

#### 2.6.3.2 Dependency on State

None.
2.6.3.3 Effect on State
None.

2.6.3.4 Errors
Table 2-12: Error Codes for GetAllowedValues()

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>711</td>
<td>Invalid DataTypeID</td>
<td>An invalid value has been specified in the DataTypeID input argument.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>Related State Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>OUT</td>
<td>StateUpdateID</td>
</tr>
</tbody>
</table>

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

2.6.4.2 Dependency on State
None.

2.6.4.3 Effect on State
None.

2.6.4.4 Errors
Table 2-14: Error Codes for GetStateUpdateID()

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
</tbody>
</table>
2.6.5  **BrowseRecordSchedules()**

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

### 2.6.5.1  Arguments

#### Table 2-15: Arguments for `BrowseRecordSchedules()`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter</td>
<td>IN</td>
<td>A_ARG_TYPE_PropertyList</td>
</tr>
<tr>
<td>StartingIndex</td>
<td>IN</td>
<td>A_ARG_TYPE_Index</td>
</tr>
<tr>
<td>RequestedCount</td>
<td>IN</td>
<td>A_ARG_TYPE_Count</td>
</tr>
<tr>
<td>SortCriteria</td>
<td>IN</td>
<td>A_ARG_TYPE_SortCriteria</td>
</tr>
<tr>
<td>Result</td>
<td>OUT</td>
<td>A_ARG_TYPE_RecordSchedule</td>
</tr>
<tr>
<td>NumberReturned</td>
<td>OUT</td>
<td>A_ARG_TYPE_Count</td>
</tr>
<tr>
<td>TotalMatches</td>
<td>OUT</td>
<td>A_ARG_TYPE_Count</td>
</tr>
<tr>
<td>UpdateID</td>
<td>OUT</td>
<td>StateUpdateID</td>
</tr>
</tbody>
</table>

#### 2.6.5.1.1  **Filter**

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

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

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

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

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

Examples of valid `Filter` argument values include:

---

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
A compliant ScheduledRecording service implementation MUST also ignore optional properties requested in the `Filter` input argument which are not actually present in the matching objects. For example, a `BrowseRecordSchedules()` `Filter` input argument of the form “srs:activePeriod” is successful and returns a `Result` value that complies with the other `BrowseRecordSchedules()` input arguments, even in the case where the objects represented in the `Result` argument do not have an `activePeriod` property defined.

### 2.6.5.1.2 StartingIndex and RequestedCount

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

### 2.6.5.1.3 SortCriteria

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

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

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

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

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

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

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

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

---

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
When a `SortCriteria` argument contains property names of optional and/or multi-valued properties, the following rules apply:

If the property is prefixed by “+” then:

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

If the property is prefixed by “-” then:

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

**Example:**

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

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

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

```
"+srs:scheduledStartDateTime"
```

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

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

because:

- `<item id="4"/>` has no `srs:scheduledStartDateTime` property, it therefore appears first.
• <item id="2"/> srs:scheduledStartDateTime property resolves to Wednesday, 2005-06-22T15:30:00 since this is the earliest date&time in the list. It therefore appears second.

• <item id="3"/> srs:scheduledStartDateTime property resolves to Wednesday, 2005-06-22T16:00:00. It therefore appears third.

• <item id="1"/> srs:scheduledStartDateTime property resolves to Tuesday, 2006-02-07T15:30:00. It therefore appears last.

Sorting on ContentDirectory service imported properties is not supported.

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

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

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

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

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

2.6.5.2 Dependency on State
None.

2.6.5.3 Effect on State
None.

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

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

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

2.6.6 `BrowseRecordTasks()`

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

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

2.6.6.1 Arguments

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RecordScheduleID</code></td>
<td>IN</td>
<td><code>A_ARG_TYPE_ObjectID</code></td>
</tr>
<tr>
<td><code>Filter</code></td>
<td>IN</td>
<td><code>A_ARG_TYPE_PropertyList</code></td>
</tr>
<tr>
<td><code>StartingIndex</code></td>
<td>IN</td>
<td><code>A_ARG_TYPE_Index</code></td>
</tr>
<tr>
<td><code>RequestedCount</code></td>
<td>IN</td>
<td><code>A_ARG_TYPE_Count</code></td>
</tr>
<tr>
<td><code>SortCriteria</code></td>
<td>IN</td>
<td><code>A_ARG_TYPE_SortCriteria</code></td>
</tr>
<tr>
<td><code>Result</code></td>
<td>OUT</td>
<td><code>A_ARG_TYPE_RecordTask</code></td>
</tr>
<tr>
<td><code>NumberReturned</code></td>
<td>OUT</td>
<td><code>A_ARG_TYPE_Count</code></td>
</tr>
<tr>
<td><code>TotalMatches</code></td>
<td>OUT</td>
<td><code>A_ARG_TYPE_Count</code></td>
</tr>
<tr>
<td><code>UpdateID</code></td>
<td>OUT</td>
<td><code>StateUpdateID</code></td>
</tr>
</tbody>
</table>

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

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

2.6.6.1.2  **Filter**

See Section 2.6.5.1.1, “Filter”.

2.6.6.1.3  **StartingIndex and RequestedCount**

See Section 2.6.5.1.2, “StartingIndex and RequestedCount”.

2.6.6.1.4  **SortCriteria**

See Section 2.6.5.1.3, “SortCriteria”.

2.6.6.1.5  **Result**

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

2.6.6.1.6  **NumberReturned**

See Section 2.6.5.1.5, “NumberReturned”.

2.6.6.1.7  **TotalMatches**

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

2.6.6.1.8  **UpdateID**

See Section 2.6.5.1.7, “UpdateID”.

2.6.6.2  **Dependency on State**

None.

2.6.6.3  **Effect on State**

None.

2.6.6.4  **Errors**

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

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
</tbody>
</table>

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>704</td>
<td>No such recordSchedule ID</td>
<td>The specified recordSchedule does not exist.</td>
</tr>
<tr>
<td>709</td>
<td>Unsupported or invalid sort criteria</td>
<td>The sort criteria specified is not supported or is invalid.</td>
</tr>
<tr>
<td>720</td>
<td>Cannot process the request</td>
<td>Cannot process the request in a reasonable amount of time.</td>
</tr>
</tbody>
</table>

### 2.6.7 CreateRecordSchedule()

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

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

### 2.6.7.1 Arguments

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elements</td>
<td>IN</td>
<td>A_ARG_TYPE_RecordScheduleParts</td>
</tr>
<tr>
<td>RecordScheduleID</td>
<td>OUT</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
<tr>
<td>Result</td>
<td>OUT</td>
<td>A_ARG_TYPE_RecordSchedule</td>
</tr>
<tr>
<td>UpdateID</td>
<td>OUT</td>
<td>StateUpdateID</td>
</tr>
</tbody>
</table>

#### 2.6.7.1.1 Elements

The Elements input argument contains an XML escaped srs XML Document (see [SRS-XSD]). This document contains a single recordScheduleParts. The recordScheduleParts object identifies the desired property values for the recordSchedule object to be created. The new recordSchedule will be an instance of a specific recordSchedule class. Each class defines its set of member properties, some of which are REQUIRED, and some of which are OPTIONAL. See Appendix C, “AV Working Committee Class Definitions” for details. All REQUIRED member properties MUST be specified. If a control point omits supported OPTIONAL member properties from the Elements argument, then the ScheduledRecording service MUST create the recordSchedule with the appropriate default value for those omitted member properties. If unsupported properties or unknown properties are specified in the Elements argument, the
ScheduledRecording service MUST gracefully accept these. If an unsupported value is specified for a supported member property, the ScheduledRecording service MUST detect this and return error code 703.

### 2.6.7.1.2 RecordScheduleID

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

### 2.6.7.1.3 Result

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

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

### 2.6.7.1.4 UpdateID

See Section 2.6.5.1.7, “UpdateID”.

### 2.6.7.2 Dependency on State

None.

### 2.6.7.3 Effect on State

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

### 2.6.7.4 Errors

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

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>701</td>
<td>Invalid Syntax</td>
<td>The recordSchedule in the Elements argument has invalid syntax. This includes malformed XML in the Elements input argument or a general schema violation.</td>
</tr>
</tbody>
</table>
## 2.6.8 **DeleteRecordSchedule()**

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

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

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

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

Table 2-21: Arguments for DeleteRecordSchedule()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordScheduleID</td>
<td>IN</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
</tbody>
</table>

2.6.8.1.1 RecordScheduleID

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

2.6.8.2 Dependency on State

None.

2.6.8.3 Effect on State

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

2.6.8.4 Errors

Table 2-22: Error Codes for DeleteRecordSchedule()

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

2.6.9 GetRecordSchedule()

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

2.6.9.1 Arguments

Table 2-23: Arguments for GetRecordSchedule()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordScheduleID</td>
<td>IN</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
<tr>
<td>Filter</td>
<td>IN</td>
<td>A_ARG_TYPE_PropertyList</td>
</tr>
<tr>
<td>Result</td>
<td>OUT</td>
<td>A_ARG_TYPE_RecordSchedule</td>
</tr>
<tr>
<td>UpdateID</td>
<td>OUT</td>
<td>StateUpdateID</td>
</tr>
</tbody>
</table>
2.6.9.1.1 **RecordScheduleID**
The `RecordScheduleID` contains the object ID of the `recordSchedule` for which information is to be returned.

2.6.9.1.2 **Filter**
See Section 2.6.5.1.1, “Filter”.

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

2.6.9.1.4 **UpdateID**
See Section 2.6.5.1.7, “UpdateID”.

2.6.9.2 **Dependency on State**
None.

2.6.9.3 **Effect on State**
None.

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

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>704</td>
<td>No such recordSchedule ID</td>
<td>The specified <code>recordSchedule</code> does not exist.</td>
</tr>
<tr>
<td>720</td>
<td>Cannot process the request</td>
<td>Cannot process the request in a reasonable amount of time.</td>
</tr>
</tbody>
</table>

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

The invocation of the `EnableRecordSchedule()` action enables all the associated `recordTask` objects in the “IDLE” or “ACTIVE” phase (See Section 2.6.14, “EnableRecordTask()”) except for those which were disabled individually at the `recordTask` level via the `DisableRecordTask()` action. Disabling at the `recordTask` level always takes precedence. If any of the associated `recordTask` objects end up in a state that indicates that these `recordTask` objects should already be recording, then the ScheduledRecording service MUST ensure that these recordings start as soon as possible and within a reasonable amount of time (this will most likely result in a partial recording). If a ScheduledRecording service implementation can not ensure that these recordings start as soon as possible, then the `EnableRecordSchedule()` action MUST...
return with error code 720. If the ScheduledRecording service can not enable some of the recordTask objects in the “IDLE” or “ACTIVE” phase, it MUST return error code 740 without any change.

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

2.6.10.1 Arguments

Table 2-25: Arguments for EnableRecordSchedule()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordScheduleID</td>
<td>IN</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
</tbody>
</table>

2.6.10.1.1 RecordScheduleID

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

2.6.10.2 Dependency on State

None.

2.6.10.3 Effect on State

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

2.6.10.4 Errors

Table 2-26: Error Codes for EnableRecordSchedule()

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

2.6.11 DisableRecordSchedule()

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

The invocation of the DisableRecordSchedule() action disables all associated recordTask objects in the “IDLE” phase (See Section 2.6.15, “DisableRecordTask()”) except for those which were enabled individually at the recordTask level via the EnableRecordTask() action. Enabling at the recordTask level
always takes precedence. If the ScheduledRecording service can not disable some of the recordTask objects in the “IDLE” phase, it MUST return error code 739 without any change.

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

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

### 2.6.11.1 Arguments

Table 2-27: Arguments for DisableRecordSchedule()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordScheduleID</td>
<td>IN</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
</tbody>
</table>

#### 2.6.11.1.1 RecordScheduleID

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

#### 2.6.11.2 Dependency on State

None.

#### 2.6.11.3 Effect on State

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

#### 2.6.11.4 Errors

Table 2-28: Error Codes for DisableRecordSchedule()

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

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

### 2.6.12.1 Arguments

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordTaskID</td>
<td>IN</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
</tbody>
</table>

#### 2.6.12.1.1 **RecordTaskID**

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

### 2.6.12.2 Dependency on State

None.

### 2.6.12.3 Effect on State

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

### 2.6.12.4 Errors

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

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>713</td>
<td>No such <strong>recordTask</strong> ID</td>
<td>The specified <strong>recordTask</strong> does not exist.</td>
</tr>
</tbody>
</table>

2.6.13 **GetRecordTask()**

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

### 2.6.13.1 Arguments

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordTaskID</td>
<td>IN</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
<tr>
<td>Filter</td>
<td>IN</td>
<td>A_ARG_TYPE_PropertyList</td>
</tr>
<tr>
<td>Result</td>
<td>OUT</td>
<td>A_ARG_TYPE_RecordTask</td>
</tr>
</tbody>
</table>
### 2.6.13.1.1 RecordTaskID
The RecordTaskID argument contains the object ID of the recordTask for which information is to be returned.

### 2.6.13.1.2 Filter
See Section 2.6.5.1.1, “Filter”.

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

### 2.6.13.1.4 UpdateID
See Section 2.6.5.1.7, “UpdateID”.

### 2.6.13.2 Dependency on State
None.

### 2.6.13.3 Effect on State
None.

### 2.6.13.4 Errors

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

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

### 2.6.14 EnableRecordTask()

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

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

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

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

### 2.6.14.1 Arguments

#### Table 2-33: Arguments for `EnableRecordTask()`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RecordTaskID</code></td>
<td>IN</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
</tbody>
</table>

#### 2.6.14.1.1 `RecordTaskID`

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

#### 2.6.14.2 Dependency on State

None.

#### 2.6.14.3 Effect on State

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

#### 2.6.14.4 Errors

#### Table 2-34: Error Codes for `EnableRecordTask()`

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>713</td>
<td>No such recordTask ID</td>
<td>The specified <code>recordTask</code> does not exist.</td>
</tr>
<tr>
<td>720</td>
<td>Cannot process the request</td>
<td>Cannot process the request in a reasonable amount of time.</td>
</tr>
<tr>
<td>741</td>
<td>recordTask in “DONE” phase</td>
<td>A <code>recordTask</code> in the “DONE” phase cannot be enabled or disabled.</td>
</tr>
</tbody>
</table>
2.6.15 **DisableRecordTask()**

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

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

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

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

### 2.6.15.1 Arguments

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RecordTaskID</code></td>
<td>IN</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
</tbody>
</table>

**2.6.15.1.1 `RecordTaskID`**

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

### 2.6.15.2 Dependency on State

None.

### 2.6.15.3 Effect on State

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

### 2.6.15.4 Errors

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

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>713</td>
<td>No such recordTask ID</td>
<td>The specified <code>recordTask</code> does not exist.</td>
</tr>
</tbody>
</table>
2.6.16 ResetRecordTask()

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

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

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

2.6.16.1 Arguments

Table 2-37: Arguments for ResetRecordTask()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordTaskID</td>
<td>IN</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
</tbody>
</table>

2.6.16.1.1 RecordTaskID

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

2.6.16.2 Dependency on State

None.

2.6.16.3 Effect on State

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

2.6.16.4 Errors

Table 2-38: Error Codes for ResetRecordTask()

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>713</td>
<td>No such recordTask ID</td>
<td>The specified recordTask does not exist.</td>
</tr>
</tbody>
</table>

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
2.6.17 GetRecordScheduleConflicts()

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

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

2.6.17.1 Arguments

Table 2-39: Arguments for GetRecordScheduleConflicts()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordScheduleID</td>
<td>IN</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
<tr>
<td>RecordScheduleConflictIDList</td>
<td>OUT</td>
<td>A_ARG_TYPE_ObjectIDList</td>
</tr>
<tr>
<td>UpdateID</td>
<td>OUT</td>
<td>StateUpdateID</td>
</tr>
</tbody>
</table>

2.6.17.1.1 RecordScheduleID

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

2.6.17.1.2 RecordScheduleConflictIDList

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

2.6.17.1.3 UpdateID

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

2.6.17.2 Dependency on State

None.

2.6.17.3 Effect on State

None.
2.6.17.4 Errors

Table 2-40: Error Codes for GetRecordScheduleConflicts()

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

2.6.18 GetRecordTaskConflicts()

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

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

2.6.18.1 Arguments

Table 2-41: Arguments for GetRecordTaskConflicts()

<table>
<thead>
<tr>
<th>Argument</th>
<th>Direction</th>
<th>relatedStateVariable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordTaskID</td>
<td>IN</td>
<td>A_ARG_TYPE_ObjectID</td>
</tr>
<tr>
<td>RecordTaskConflictIDList</td>
<td>OUT</td>
<td>A_ARG_TYPE_ObjectIDList</td>
</tr>
<tr>
<td>UpdateID</td>
<td>OUT</td>
<td>StateUpdateID</td>
</tr>
</tbody>
</table>

2.6.18.1.1 RecordTaskID

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

2.6.18.1.2 RecordTaskConflictIDList

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

2.6.18.1.3 UpdateID

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

2.6.18.2 Dependency on State

None.
2.6.18.3 Effect on State

None.

2.6.18.4 Errors

Table 2-42: Error Codes for GetRecordTaskConflicts()

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

2.6.19 Common Error Codes

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

Table 2-43: Common Error Codes

<table>
<thead>
<tr>
<th>ErrorCode</th>
<th>errorDescription</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400-499</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>500-599</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>600-699</td>
<td>TBD</td>
<td>See UPnP Device Architecture section on Control.</td>
</tr>
<tr>
<td>700</td>
<td>Reserved for future extensions.</td>
<td></td>
</tr>
<tr>
<td>701</td>
<td>Invalid Syntax</td>
<td>The recordSchedule in the Elements argument has invalid syntax. This includes malformed XML in the Elements input argument or a general schema violation.</td>
</tr>
<tr>
<td>702</td>
<td>Reserved for future extensions.</td>
<td></td>
</tr>
<tr>
<td>703</td>
<td>Invalid Value</td>
<td>One or more properties in the input recordSchedule (in the Elements argument) have an invalid value.</td>
</tr>
<tr>
<td>704</td>
<td>No such recordSchedule ID</td>
<td>The specified recordSchedule does not exist.</td>
</tr>
<tr>
<td>705</td>
<td>Active recordTask</td>
<td>One or more recordTask instances are actively recording.</td>
</tr>
<tr>
<td>706</td>
<td>Reserved for future extensions.</td>
<td></td>
</tr>
<tr>
<td>707</td>
<td>Read-only property</td>
<td>Unable to specify read-only property.</td>
</tr>
<tr>
<td>708</td>
<td>Required property</td>
<td>Omitting a REQUIRED property is not allowed</td>
</tr>
<tr>
<td>709</td>
<td>Unsupported or invalid sort criteria</td>
<td>The sort criteria specified are not supported or are invalid.</td>
</tr>
<tr>
<td>710</td>
<td>Reserved for future extensions.</td>
<td></td>
</tr>
<tr>
<td>ErrorCode</td>
<td>ErrorDescription</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>711</td>
<td>Invalid DataTypeID</td>
<td>An invalid value has been specified in the <code>DataTypeID</code> input argument.</td>
</tr>
<tr>
<td>712</td>
<td></td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>713</td>
<td>No such recordTask ID</td>
<td>The specified <code>recordTask</code> does not exist.</td>
</tr>
<tr>
<td>714-719</td>
<td></td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>720</td>
<td>Cannot process the request</td>
<td>Cannot process the request in a reasonable amount of time.</td>
</tr>
<tr>
<td>721-729</td>
<td></td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>730</td>
<td>Conflict</td>
<td>The specified <code>recordSchedule</code> is conflicting with one or more existing <code>recordSchedule</code> objects. The ScheduledRecording service MAY reject a conflicting <code>recordSchedule</code> and return with this error code.</td>
</tr>
<tr>
<td>731</td>
<td>Protected Contents</td>
<td>The specified contents are copy protected. The ScheduledRecording service MAY reject a <code>recordSchedule</code> that specifies copy protected contents and return with this error code.</td>
</tr>
<tr>
<td>732</td>
<td>No Media</td>
<td>The specified removable media is not inserted.</td>
</tr>
<tr>
<td>733</td>
<td>Media Write Protect</td>
<td>The specified removable media is write-protected.</td>
</tr>
<tr>
<td>734</td>
<td>Media No Space</td>
<td>The specified media does not have sufficient capacity.</td>
</tr>
<tr>
<td>735</td>
<td>Media Error</td>
<td>Error related to the specified destination media.</td>
</tr>
<tr>
<td>736</td>
<td>Too many record schedules</td>
<td>The maximum number of <code>recordSchedule</code> objects is reached.</td>
</tr>
<tr>
<td>737</td>
<td>Resource Error</td>
<td>Error related to an application resource.</td>
</tr>
<tr>
<td>738</td>
<td></td>
<td>Reserved for future extensions.</td>
</tr>
<tr>
<td>739</td>
<td>Cannot enable/disable recordSchedule</td>
<td>One or more of the associated <code>recordTask</code> objects could not be enabled or disabled.</td>
</tr>
<tr>
<td>740</td>
<td>recordSchedule “COMPLETED”</td>
<td>The <code>recordSchedule</code> has already completed and cannot be enabled or disabled.</td>
</tr>
<tr>
<td>741</td>
<td>recordTask in “DONE” phase</td>
<td>A <code>recordTask</code> in the “DONE” phase cannot be enabled or disabled.</td>
</tr>
</tbody>
</table>

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

### 2.7 State Diagram of `recordTask`

In the ScheduledRecording service, the state of each `recordTask` is represented by its state properties (that is: `taskState` and its associated properties `taskState@xxx`). The definitions are described in Appendix B.16, “Task State Properties”. Additionally, the state behavior of a `recordTask` is illustrated by a state diagram to give a visual description of each state and the state transitions. State diagrams are provided for
informational purposes. Whenever there is a discrepancy between the state diagram and the textual description of state and state transition, the normative textual description takes precedence.

### 2.7.1 A Full-Featured State Diagram

As described above, the `taskState` property reflects the current state of the `recordTask`. Its value changes over time as the `recordTask` progresses through its life-cycle. The following state transition diagram shows the possible states and state transitions that a given `recordTask` may take throughout its life time. It is assumed that all (REQUIRED and OPTIONAL) normative states and attributes of a `recordTask` are supported by the device. Further, it is assumed that a device is able to resume recording in the middle of the “ACTIVE” phase. The `GetAllowedValues()` action can be used to determine if a device supports all states and attributes.
Figure 5: A Full-Featured State Diagram
2.7.1.1 “IDLE” Phase
The states in this phase indicate that the recordTask’s start time has not yet been reached and that the target content is not yet available for recording. The recordTask will remain in the “IDLE” phase (that is: in any of the IDLE states), until either the recordTask’s start time is reached or a fatal error is detected. If/when the start time is reached, the recordTask will transition to one of the states in the “ACTIVE” phase. If a fatal error is detected, the recordTask will transition directly to the “DONE.EMPTY” state within the “DONE” phase.

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

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

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

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

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

### 2.7.1.2.2 “**ACTIVE.TRANSITION.RESTART**” State

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

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

### 2.7.1.2.3 “**ACTIVE.RECORDING.FROMSTART.OK**” State

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

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

### 2.7.1.2.4 “**ACTIVE.RECORDING.FROMSTART.ATRISK**” State

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

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

2.7.1.2.5 “ACTIVE RECORDING RESTART OK” State

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

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

2.7.1.2.6 “ACTIVE RECORDING RESTART AT RISK” State

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

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

2.7.1.2.7 “ACTIVE NOTRECORDING” State

This state indicates that a non-fatal error has occurred while the device was recording the target content or while the device was attempting to start recording the target content. The `recordTask` remains in this state
until either the recordTask’s end time is reached or until all of the current non-fatal errors are resolved, or until a fatal error actually occurs.

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

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

2.7.1.3.1 “DONE.FULL” State
This state indicates that all of the recordTask’s target content was recorded in its entirety without any interruptions. No error occurred while recording the target content. The recordTask remains in this state until the recordTask is deleted.

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

2.7.1.3.3 “DONE.EMPTY” State
This state indicates that none of the recordTask’s target content was recorded. One or more errors occurred that prevented the recording from even getting started. The recordTask remains in this state until the recordTask is deleted.

2.7.2 A Minimal-Implementation State Diagram
The simplest state diagram based on the minimum required state related properties is illustrated below to show the behavior of such a device and the progression of its state. The support level of these state related properties is defined in Appendix C.3.2, “object.recordTask Class”. This example only uses the set of REQUIRED allowed values for the taskState property. In the example below, it is assumed that the device is UNABLE to resume recording once the “ACTIVE” phase is entered. By definition, any device MUST support at least the following 5 illustrated states.
Figure 6: A Minimal-Implementation State Diagram
2.7.2.1 **“IDLE” Phase**

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

2.7.2.1.1 **“IDLE.READY” State**

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

2.7.2.1.2 **“IDLE.ATRISK” State**

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

2.7.2.2 **“ACTIVE” Phase**

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

2.7.2.2.1 **“ACTIVE.TRANSITION.FROMSTART” State**

This state is not supported.

2.7.2.2.2 **“ACTIVE.TRANSITION.RESTART” State**

This state is not supported.

2.7.2.2.3 **“ACTIVE.RECORDING.FROMSTART.OK” State**

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

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

2.7.2.2.4 **“ACTIVE.RECORDING.FROMSTART.ATRISK” State**

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

2.7.2.2.5 **“ACTIVE.RECORDING.RESTART.OK” State**

This state is not supported since the device can not resume an interrupted recording or catch a late recording that misses the beginning.
2.7.2.2.6 “ACTIVE.RECORDING.RESTART.ATRISK” State
This state is not supported since the device does not support pending errors.

2.7.2.2.7 “ACTIVE.NOTRECORDING” State
This state is not supported since the device can not resume interrupted recording. Any interruptions during the middle of recording will cause a transition to the “DONE” phase.

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

2.7.2.3.1 “DONE.FULL” State
This state indicates a perfect recording. The recordTask’s target content was recorded in its entirety without any interruptions.

2.7.2.3.2 “DONE.PARTIAL” State
This state indicates that part of the recordTask’s target content was recorded, but not all of it. This state is reached from an “ACTIVE” recordTask due to a fatal error.

2.7.2.3.3 “DONE.EMPTY” State
This state indicates that none of the recordTask’s target content was recorded. It is a result of a recording that has never been started due to a fatal error.

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

The events occurs at:
- T0: Sytem is idle.
- T1: Error 1 (for example, DRM protected is being broadcast) and Error 3 (for example, conflicted-loser) are predicted.
- T2: The recordSchedule reaches the scheduled start time, but Error 1 prevents the recording from starting.
- T3: Suddenly, a new Error 2 occurs (for example, disabled)
- T4: Error 1 is fixed (for example, the protected part ends.), but Error 3 is still predicted.
- T5: Error 2 is fixed (for example, enabled by user), but Error 3 is still predicted.
- T6: Error 3 occurs (for example, other prioritized program starts)
- T7: Error 3 is fixed (for example, the prioritized program ends)
- T8: The recordSchedule reached the scheduled end time
### Table 2-44: recordTask State Timeline

<table>
<thead>
<tr>
<th>Recording Schedule Time Line</th>
<th>Error 1</th>
<th>Error 2</th>
<th>Error 3</th>
<th>taskState</th>
<th>@phase</th>
<th>@recording</th>
<th>@someBitsRecorded</th>
<th>@someBitsMissing</th>
<th>@startDateTimeMet</th>
<th>@firstBitsRecorded</th>
<th>@lastBitsRecorded</th>
<th>@fatalError</th>
<th>@currentErrors</th>
<th>@pendingErrors</th>
<th>@errorHistory</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>&quot;IDLE.READY&quot;</td>
<td>&quot;IDLE&quot;</td>
<td></td>
<td>0 0 0 0 0 0 0 0 Ø Ø Ø</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>&quot;IDLE.ATRISK&quot;</td>
<td>&quot;IDLE&quot;</td>
<td></td>
<td>0 0 0 0 0 0 0 0 Ø 1,3 Ø</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>&quot;ACTIVE.NOTRECORDING&quot;</td>
<td>&quot;ACTIVE&quot;</td>
<td></td>
<td>0 0 1 1 0 0 0 0 1 3 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>&quot;ACTIVE.NOTRECORDING&quot;</td>
<td>&quot;ACTIVE&quot;</td>
<td></td>
<td>0 0 1 1 0 0 0 0 1,2 3 1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>&quot;ACTIVE.NOTRECORDING&quot;</td>
<td>&quot;ACTIVE&quot;</td>
<td></td>
<td>0 0 1 1 0 0 0 0 2 3 1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>&quot;ACT.RECORDING.RESTART.ATRISK&quot;</td>
<td>&quot;ACTIVE&quot;</td>
<td></td>
<td>1 1 1 1 0 0 0 0 Ø 3 1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td>&quot;ACTIVE.NOTRECORDING&quot;</td>
<td>&quot;ACTIVE&quot;</td>
<td></td>
<td>0 1 1 1 0 0 0 0 12,3 Ø</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T7</td>
<td>&quot;ACT.RECORDING.RESTART.OK&quot;</td>
<td>&quot;ACTIVE&quot;</td>
<td></td>
<td>1 1 1 1 0 0 0 0 Ø Ø 1,2,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T8</td>
<td>&quot;DONE.PARTIAL&quot;</td>
<td>&quot;DONE&quot;</td>
<td></td>
<td>0 1 1 1 1 0 1 0 Ø Ø 1,2,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2.8 ScheduledRecording Service Priority Model

#### 2.8.1 Introduction of the ScheduledRecording Service Priority Model

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

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

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

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

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

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
The list of priority levels supported by a ScheduledRecording service is obtainable via the GetAllowedValues() action by examining the allowed value list of the priority property. Each existing recordSchedule (on a given ScheduledRecording service) MUST be assigned one of these supported priority levels. The priority property of each recordSchedule indicates the current priority level assigned to that recordSchedule which can be retrieved via the BrowseRecordSchedules() action.

2.8.2 Ordered Priority within Each Priority Level

In addition to supporting one or more priority levels, some ScheduledRecording service implementations are able to prioritize the recordSchedule instances within each priority level. When ordered priority is supported, each recordSchedule (in addition to its assigned priority level) is also assigned a unique “ordered priority slot” ranging from 1 to <n> where <n> is the total number of recordSchedule instances within the ScheduledRecording service. A value of 1 represents the highest priority recordSchedule within the ScheduledRecording service; that is: the highest priority recordSchedule within the highest priority level “L1”. The value <n> represents the lowest priority recordSchedule within the ScheduledRecording service; that is: the lowest priority recordSchedule within the lowest priority level “L5”. The ordered priority slot assigned to each recordSchedule can be obtained via the recordSchedule’s priority@orderedValue property. A ScheduledRecording service that support ordered priority MUST expose this property for each of their recordSchedule instances. Conversely, a ScheduledRecording service that does not support this capability MUST NOT expose the priority@orderedValue property. Within a given ScheduledRecording service, each ordered priority slot is assigned to exactly one recordSchedule.

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

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

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

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

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>RecordScheduleID</th>
<th>Ordered Priority Slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>“L1” (highest priority level)</td>
<td>RS-A</td>
<td>1</td>
</tr>
<tr>
<td>“L2”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“L3”</td>
<td>RS-C</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>RS-B</td>
<td>3</td>
</tr>
<tr>
<td>“L4”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“L5” (lowest priority level)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 2-46: Example 2: More recordSchedule instances than the Number of Supported Priority Levels.

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>RecordScheduleID</th>
<th>Ordered Priority Slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>“L1” (highest priority level)</td>
<td>RS-A</td>
<td>1</td>
</tr>
<tr>
<td>“L2”</td>
<td>RS-F</td>
<td>2</td>
</tr>
<tr>
<td>“L3”</td>
<td>RS-C, RS-B</td>
<td>3, 4</td>
</tr>
<tr>
<td>“L4”</td>
<td>RS-E</td>
<td>5</td>
</tr>
<tr>
<td>“L5” (lowest priority level)</td>
<td>RS-G, RS-D</td>
<td>6, 7</td>
</tr>
</tbody>
</table>

#### 2.8.3 Setting the Initial Priority Level of a recordSchedule

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

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

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

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

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

Additionally, when ordered priority is supported, the ScheduledRecording service MUST also support a number of convenience values corresponding to the highest and lowest ordered priority slots within each of its supported priority level. These convenience values MUST have the form “L<><_HI” or “L<><_LOW” where “L” is an abbreviation for “Level”, <x> is a number ranging from 1 to some device-specific maximum value n where n is the total number of distinct priority levels supported by the ScheduledRecording service. For example, a ScheduledRecording service that supports 5 priority levels and also ordered priority MUST support the values “L1_HI”, “L1_LOW”, “L2_HI”, “L2_LOW”, “L3_HI”,
“L3 LOW”, “L4 HI”, “L4 LOW”, “L5 HI”, “L5 LOW” for the desiredPriority property. Furthermore, the ScheduledRecording service MUST also support two additional convenience values corresponding to the highest and lowest priority within the ScheduledRecording service. These two additional convenience values are “HIGHEST” (which is equivalent to the highest ordered priority slot in the highest priority level “L1 HI”), and “LOWEST” (which is equivalent to the lowest priority slot within the lowest priority level “L<n> LOW” when n is the total number of priority slots supported by the ScheduledRecording service).

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

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

Table 2-47: Existing recordSchedule Priorities

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>RecordScheduleID</th>
<th>Ordered Priority Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“L1” (highest priority level)</td>
<td>RS-A</td>
<td>1</td>
</tr>
<tr>
<td>“L2”</td>
<td>RS-C</td>
<td>2</td>
</tr>
<tr>
<td>“L3” (lowest priority level)</td>
<td>RS-B</td>
<td>3</td>
</tr>
</tbody>
</table>

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

Table 2-48: desiredPriority Property Set to “RS-C”

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>RecordScheduleID</th>
<th>Ordered Priority Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“L1” (highest priority level)</td>
<td>RS-A</td>
<td>1</td>
</tr>
<tr>
<td>“L2”</td>
<td>RS-D RS-C</td>
<td>2 3</td>
</tr>
<tr>
<td>“L3” (lowest priority level)</td>
<td>RS-B</td>
<td>4</td>
</tr>
</tbody>
</table>

Next the CreateRecordSchedule() action is invoked with the desiredPriority property set to “HIGHEST”, “L1 HI”, or “RS-A” (all values have the same effect). After the action completes, a new recordSchedule is created with the @id property set to “RS-E”. The set of recordSchedule instances is now prioritized as follows:

Table 2-49: desiredPriority Property Set to “HIGHEST”, “L1 HI”, or “RS-A”

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>RecordScheduleID</th>
<th>Ordered Priority Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“L1” (highest priority level)</td>
<td>RS-E RS-A</td>
<td>1 2</td>
</tr>
<tr>
<td>“L2”</td>
<td>RS-D RS-C</td>
<td>3 4</td>
</tr>
<tr>
<td>“L3” (lowest priority level)</td>
<td>RS-B</td>
<td>5</td>
</tr>
</tbody>
</table>

Now the CreateRecordSchedule() action is invoked with the desiredPriority property set to “LOWEST”, “L3 LOW”, or “RS-B” (all values have the same effect). After the action completes, a new recordSchedule...
is created with the @id property set to “RS-F”. The set of recordSchedule instances is now prioritized as follows:

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

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>RecordScheduleID</th>
<th>Ordered Priority Value</th>
</tr>
</thead>
</table>
| “L1” (highest priority level) | RS-E  
RS-A | 1  
2 |
| “L2”         | RS-D  
RS-C | 3  
4 |
| “L3” (lowest priority level) | RS-B  
RS-F | 5  
6 |

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

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

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>RecordScheduleID</th>
<th>Ordered Priority Value</th>
</tr>
</thead>
</table>
| “L1” (highest priority level) | RS-E  
RS-A | 1  
2 |
| “L2”         | RS-D  
RS-G  
RS-C | 3  
4  
5 |
| “L3” (lowest priority level) | RS-B  
RS-F | 6  
7 |

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

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

**2.9 Theory of Operation**

**2.9.1 Introduction**

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

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

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

2.9.2.1 Checking the Sort Capabilities

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

**Request:**
GetSortCapabilities()

will result in the following response:

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

2.9.2.2 Checking Supported Properties and their Allowed Values

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

2.9.2.2.1 Minimal Implementation Example

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

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

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

**Request:**
GetPropertyList("A_ARG_TYPE_RecordScheduleParts")

Then the following response will be generated:

**Response:**
GetPropertyList("srs:@id,srs:title,srs:class,srs:scheduledCDSObjectID, srs:scheduledStartDate,srs:scheduledDuration")

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

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

The following response will be generated:

Response:
GetAllowedValues("
<?xml version="1.0" encoding="UTF-8"?>
<AVDT
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:srs="urn:schemas-upnp-org:av:srs"
 xmlns="urn:schemas-upnp-org:av:avdt"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
  urn:schemas-upnp-org:av:srs
  http://www.upnp.org/schemas/av/srs.xsd
  urn:schemas-upnp-org:av:avdt
  http://www.upnp.org/schemas/av/avdt.xsd"
<contextID>uuid:device-UUID::urn:schemas-upnp-org:service:ScheduledRecording:1</contextID>
<dataStructType>A_ARG_TYPE_RecordScheduleParts</dataStructType>

<fieldTable>
  <field>
    <name>srs:@id</name>
    <dataType>xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
      <allowAny/></allowAny>
    </allowedValueDescriptor>
  </field>
  
  <field>
    <name>srs:title</name>
    <dataType>xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
      <allowAny/></allowAny>
    </allowedValueDescriptor>
  </field>
  
  <field>
    <name>srs:class</name>
    <dataType>xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
      <allowedValueList>
        <allowedValue>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</allowedValue>
      </allowedValueList>
    </allowedValueDescriptor>
  </field>
  
  <field>
    <name>srs:scheduledCDSObjectID</name>
    <dataType>xsd:string</dataType>
  </field>
</fieldTable>
Assume further that the control point wants to determine which properties it can expect to get returned in the Result output argument of the CreateRecordSchedule() action of that same minimal ScheduledRecording implementation. It issues the following request (The Result argument of the CreateRecordSchedule() action is of data type A_ARG_TYPE_RecordSchedule):

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

**Request:**
GetPropertyList("A_ARG_TYPE_RecordSchedule")

The following response will be generated:

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

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

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

**Request:**
GetAllowedValues("A_ARG_TYPE_RecordSchedule",

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
The following response will be generated:

Response:
GetAllowedValues("srs:recordDestination,srs:recordDestination@mediaType,srs:scheduleState,srs:scheduleState@currentErrors,srs:abnormalTasksExist,srs:currentRecordTaskCount")

<?xml version="1.0" encoding="UTF-8"?>
<AVDT xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:srs="urn:schemas-upnp-org:av:srs"
     xmlns="urn:schemas-upnp-org:av:avdt"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:schemas-upnp-org:av:srs
     http://www.upnp.org/schemas/av/srs.xsd
     urn:schemas-upnp-org:av:avdt
     http://www.upnp.org/schemas/av/avdt.xsd">
  <contextID>uuid:device-UUID::urn:schemas-upnp-org:service:ScheduledRecording:1</contextID>
  <dataStructType>A_ARG_TYPE_RecordSchedule</dataStructType>
  <fieldTable>
    <field>
      <name>srs:recordDestination</name>
      <dataType>xsd:string</dataType>
      <minCountTotal>1</minCountTotal>
      <allowedValueDescriptor>
        <allowedValueList>
          <allowedValue>Hard Disk</allowedValue>
        </allowedValueList>
      </allowedValueDescriptor>
    </field>
    <field>
      <name>srs:recordDestination@mediaType</name>
      <dataType>xsd:string</dataType>
      <allowedValueDescriptor>
        <dependentField>
          <name>srs:recordDestination</name>
          <anyValue></anyValue>
        </dependentField>
        <minCount>1</minCount>
        <allowedValueList>
          <allowedValue>HDD</allowedValue>
        </allowedValueList>
      </allowedValueDescriptor>
    </field>
    <field>
      <name>srs:scheduleState</name>
      <dataType>xsd:string</dataType>
      <minCountTotal>1</minCountTotal>
      <allowedValueDescriptor>
        <allowedValueList>
          <allowedValue>OPERATIONAL</allowedValue>
        </allowedValueList>
      </allowedValueDescriptor>
    </field>
  </fieldTable>
</AVDT>
Assume further that the control point wants to determine which properties it can expect to get returned in the Result output argument of the GetRecordTask() action of that same minimal ScheduledRecording implementation. It issues the following request (The Result argument of the GetRecordTask() action is of data type A_ARG_TYPE_RecordTask):

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

**Request:**
GetPropertyList("A_ARG_TYPE_RecordTask")

The following response will be generated:

**Response:**
GetPropertyList(
"srs:@id, srs:title, srs:class, srs:priority,
 srs:recordDestination, srs:recordDestination@mediaType,
..."
If the control point then wants to investigate further what values it may expect for some of those properties when browsing a \textit{recordTask}, it can retrieve that information using the following request (the Filter argument contains only a subset of the possible properties in this example):

Note: specifying "*: *" in the Filter argument is again equivalent to specifying the complete list of properties returned in the PropertyList argument of the \texttt{GetPropertyList()} action with the \texttt{DataTypeID} argument set to "A ARG TYPE RecordTask".

\textbf{Request:}
\texttt{GetAllowedValues("A ARG TYPE RecordTask", "srs:recordDestination,srs:recordDestination@mediaType, srs:taskState,srs:taskState@currentErrors")}

The following response will be generated:

\textbf{Response:}
\texttt{GetAllowedValues("<?xml version="1.0" encoding="UTF-8"?>
<AVDT
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:srs="urn:schemas-upnp-org:av:srs"
xmlns="urn:schemas-upnp-org:av:avdt"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
urn:schemas-upnp-org:av:srs
http://www.upnp.org/schemas/av/srs.xsd
urn:schemas-upnp-org:av:avdt
http://www.upnp.org/schemas/av/avdt.xsd">
<contextID>
  uuid:device-UUID::urn:schemas-upnp-org:service:ScheduledRecording:1
</contextID>
<dataStructType>A ARG TYPE RecordTask</dataStructType>

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

  <field>
    <name>srs:recordDestination@mediaType</name>
    <dataType>xsd:string</dataType>
  </field>
</fieldTable>
2.9.2.2.2 Full-fledged Implementation Example

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

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

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

**Request:**
GetPropertyList("A_ARG_TYPE_RecordScheduleParts")

Then the following response will be generated:

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

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

The following response will be generated:

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

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

Request:
GetPropertyList("A_ARG_TYPE_RecordSchedule")

The following response will be generated:

Response:
GetPropertyList(
"srs:@id,
 srs:title,
 srs:class,
 srs:additionalStatusInfo,
 srs:cdsReference,
 srs:cdsReference@link,
 srs:priority,
 srs:priority@orderedValue,
 srs:desiredPriority,
 srs:desiredPriority@type,
 srs:recordDestination,
 srs:recordDestination@mediaType,
 srs:recordDestination@targetURL,
 srs:recordDestination@preference,
 srs:desiredRecordQuality,
 srs:desiredRecordQuality@type,
 srs:scheduledCDSObjectID,
 srs:scheduledCDSObjectID@link,
 srs:scheduledChannelID,
 srs:scheduledChannelID@type,
 srs:scheduledStartDateDateTime,
 srs:scheduledDuration,
 srs:scheduledProgramCode,
 srs:scheduledProgramCode@type,
 srs:matchingName,
 srs:matchingName@type,
 srs:matchingName@substringMatch,
 srs:matchingID,
 srs:matchingID@type,
 srs:matchingChannelID,
 srs:matchingChannelID@type,
 srs:matchingStartDateDateTimeRange,
 srs:matchingDurationRange,
 srs:matchingRatingLimit,
 srs:matchingRatingLimit@type,
 srs:matchingEpisodeType,
 srs:totalDesiredRecordTasks,
If the control point then wants to investigate further what values it may expect for all of those properties when browsing a recordSchedule, it can retrieve that information using the following (The Result output argument of the CreateRecordSchedule() action is of data type A_ARG_TYPE_RecordSchedule):

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

The following response will be generated:

Response:
See Appendix G.1, “A_ARG_TYPE_RecordSchedule AVDT Example” for a complete response message.

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

Request:
GetPropertyList("A_ARG_TYPE_RecordTask")

The following response will be generated:

Response:
GetPropertyList(
  "srs:@id,
  srs:title,
  srs:class,
  srs:additionalStatusInfo,
  srs:cdsReference,
  srs:cdsReference@link,
  srs:priority,
  srs:priority@orderedValue,
  srs:desiredPriority,
  srs:desiredPriority@type,
  srs:recordDestination,
  srs:recordDestination@mediaType,
  srs:recordDestination@targetURL,
  srs:recordDestination@preference,
  srs:desiredRecordQuality,"
scheduledRecording:2 Service Template Version 1.01 – Document Version 1.0

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

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

The following response will be generated:

**Response:**
See Appendix G.2, “A_ARG_TYPE_RecordTask AVDT Example” for a complete response message.

### 2.9.3 Adding a Scheduled Recording Entry to the List

The following examples illustrate how to create a `recordSchedule` entry in the list of `recordSchedule` instances by invoking the `CreateRecordSchedule()` action, using the different available `recordSchedule` classes. It is assumed that the implementation used in the examples that follow supports the allowed values for the `desiredRecordQuality` and `desiredRecordQuality@type` properties as indicated in Table B-9,
“desiredRecordQuality Example” and for the recordQuality and recordQuality@type properties as indicated in Table B-33, “recordQuality Example”.

2.9.3.1 **object.recordSchedule.direct** classes

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

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

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

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

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

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

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

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

**Request:**
CreateRecordSchedule(""
<?xml version="1.0" encoding="UTF-8"?>
<srs
 xmlns="urn:schemas-upnp-org:av:srs"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:srs
 http://www.upnp.org/schemas/av/srs.xsd">
 <item id="">
    <title>BBC News at 7pm</title>
    <class>OBJECT.RECORDSCHEDULE.DIRECT_MANUAL</class>
    <desiredPriority type="PREDEF">L2</desiredPriority>
    <recordDestination mediaType="HDD" preference="1">
        Hard Disk
    </recordDestination>
</item>
</srs>"")
If the creation of the `recordSchedule` is successful, the `CreateRecordSchedule()` action returns the following `srs XML Document` in the `Result` output argument. The ScheduledRecording service MUST add unspecified supported OPTIONAL properties to convey default settings (Note that this `srs XML Document` MUST be properly escaped). One or more `recordTask` instances may be created as a result of the `recordSchedule` creation. In this example, it is assumed that 2 `recordTask` instances are spawned immediately and it is also assumed that 2 new items are created in the associated ContentDirectory service that will hold the recorded content once the recordings are made (object IDs “rec00001” and “rec00002” are assigned).

**Response:**

```xml
<CreateRecordSchedule("s101","<?xml version="1.0" encoding="UTF-8"?>
<srs xmlns="urn:schemas-upnp-org:av:srs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:schemas-upnp-org:av:srs
http://www.upnp.org/schemas/av/srs.xsd">
  <item id="s101">
    <title>BBC News at 7pm</title>
    <class>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</class>
    <priority>L2</priority>
    <desiredPriority type="PREDEF">L2</desiredPriority>
    <recordDestination mediaType="HDD" preference="1">
      Hard Disk
    </recordDestination>
    <recordDestination mediaType="DVD+R" preference="2">
      DVD Recorder
    </recordDestination>
    <desiredRecordQuality type="DEFAULT">
      SD,AUTO
    </desiredRecordQuality>
  </item>
</srs>
```
<scheduledChannelID type="ANALOG">47</scheduledChannelID>
<scheduledStartDateTime>T19:00:00</scheduledStartDateTime>
<scheduledDuration>P01:00:00</scheduledDuration>
	<totalDesiredRecordTasks>0</totalDesiredRecordTasks>
	<scheduledStartDateTimeAdjust>-P00:02:30</scheduledStartDateTimeAdjust>
	<scheduledDurationAdjust>+P00:05:00</scheduledDurationAdjust>
<activePeriod>NOW/INFINITY</activePeriod>
<persistedRecordings
latest="1"
preAllocation="0"
storedLifetime="ANY">
3
</persistedRecordings>
<scheduleState
currentErrors="">
OPERATIONAL
</scheduleState>
<abnormalTasksExist>0</abnormalTasksExist>
<currentRecordTaskCount>2</currentRecordTaskCount>
<totalCreatedRecordTasks>2</totalCreatedRecordTasks>
<totalCompletedRecordTasks>0</totalCompletedRecordTasks>
</item>
</srs>

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

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

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

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

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

**Request:**
CreateRecordSchedule("?

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rs
 xmlns="urn:schemas-upnp-org:av:rs"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:schemas-upnp-org:av:rs
 http://www.upnp.org/schemas/av/srs.xsd">
 <item id="">
  <title>UPnP Awards Ceremony</title>
  <class>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</class>
  <desiredPriority type="PREDEF">L1</desiredPriority>
  <recordDestination mediaType="HDD" preference="1">
    Hard Disk
  </recordDestination>
  <desiredRecordQuality type="QLEVEL">
    Q3
  </desiredRecordQuality>
  <scheduledCDSObjectID>
    epg_2005-04-01T09:00:00_P15:00:00
  </scheduledCDSObjectID>
  <totalDesiredRecordTasks>1</totalDesiredRecordTasks>
  <scheduledStartDateDateTimeAdjust> -P00:02:00 </scheduledStartDateDateTimeAdjust>
  <scheduledDurationAdjust> +P00:15:00 </scheduledDurationAdjust>
  <activePeriod>NOW/INFINITY</activePeriod>
  <durationLimit effect="LAST">P15:00:00</durationLimit>
  <channelMigration>1</channelMigration>
  <timeMigration>1</timeMigration>
  <persistedRecordings latest="1"
    preAllocation="0"
    storedLifetime="ANY">
    1
  </persistedRecordings>
 </item>
</rs>
```

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

**Response:**
CreateRecordSchedule("s102","

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

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
2.9.3.1.3 Creating a object.recordSchedule.direct.cdsNonEPG Class recordSchedule

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

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

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

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

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

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

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

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

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

Request:
CreateRecordSchedule("<?xml version="1.0" encoding="UTF-8"?>
<srs xmlns="urn:schemas-upnp-org:av:srs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:schemas-upnp-org:av:srs
http://www.upnp.org/schemas/av/srs.xsd">
  <item id="">
    <title>Life of a Software Developer</title>
    <class>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</class>
    <desiredPriority type="PREDEF">L1</desiredPriority>
    <recordDestination mediaType="HDD" preference="1">Hard Disk</recordDestination>
    <recordDestination mediaType="HDD" preference="2" targetURL="http://192.168.0.12/MyNAS/RecordedTV">Shared Content</recordDestination>
    <desiredRecordQuality type="DEFAULT">HD</desiredRecordQuality>
    <scheduledCDSObjectID>User_Channel_5</scheduledCDSObjectID>
    <scheduledStartDateTime>MONT19:00:00</scheduledStartDateTime>
    <scheduledDuration>F01:00:00</scheduledDuration>
  </item>
</srs>"
If the creation of the recordSchedule is successful, the CreateRecordSchedule() action returns the following srs XML Document in the Result output argument. The ScheduledRecording service MUST add unspecified supported OPTIONAL properties to convey default settings (Note that this srs XML Document MUST be properly escaped). One or more recordTask instances may be created as a result of the recordSchedule creation. In this example, it is assumed that 2 recordTask instances are spawned immediately.

Response:
  <item id="s103">
    <title>Life of a Software Developer</title>
    <class>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</class>
    <priority>L1</priority>
    <desiredPriority type="PREDEF">L1</desiredPriority>
    <recordDestination mediaType="HDD" preference="1">Hard Disk</recordDestination>
    <recordDestination mediaType="HDD" preference="2" targetURL="http://192.168.0.12/MyNAS/RecordedTV">Shared Content</recordDestination>
    <desiredRecordQuality type="DEFAULT">HD</desiredRecordQuality>
    <scheduledCDSObjectID>User_Channel_5</scheduledCDSObjectID>
    <scheduledStartDateTime>MONT19:00:00</scheduledStartDateTime>
    <scheduledDuration>P01:00:00</scheduledDuration>
  </item>
</srs>"
2.9.3.1.4 Creating a object.recordSchedule.direct.programCode Class recordSchedule

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

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

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

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

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

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

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

Request:
CreateRecordSchedule(
<?xml version="1.0" encoding="UTF-8"?>
+P00:03:00
</scheduledDurationAdjust>
<activePeriod>NOW/09-30T23:59:59</activePeriod>
<persistedRecordings
latest="1"
preAllocation="0"
storedLifetime="INFINITY">
13
</persistedRecordings>
<scheduleState
currentErrors="">
OPERATIONAL
</scheduleState>
<abnormalTasksExist>0</abnormalTasksExist>
<currentRecordTaskCount>2</currentRecordTaskCount>
<totalCreatedRecordTasks>2</totalCreatedRecordTasks>
<totalCompletedRecordTasks>0</totalCompletedRecordTasks>
</item>
</srs>"
If the creation of the recordSchedule is successful, the CreateRecordSchedule() action returns the following srs XML Document in the Result output argument. The ScheduledRecording service MUST add unspecified supported OPTIONAL properties to convey default settings (Note that this srs XML Document MUST be properly escaped). One recordTask instance may be created as a result of the recordSchedule creation. In this example, it is assumed that the recordTask instance is spawned immediately.

Response:
CreateRecordSchedule("s104"," 
<?xml version="1.0" encoding="UTF-8"?>
<srs ...

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
2.9.3.2 object.recordSchedule.query classes

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

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

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

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

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

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

The recordings must be stored on the internal Hard Disk. If, for some reason, the Hard Disk is not available at the time of recording, the recording must be canceled. The control point also specifies that these recordings should be encoded using a low record quality (“SD”). If that is not possible, medium record quality (“ED”) may also be used. If that is not possible, no recording will be made. No pre-roll or post-roll
times are specified. If the broadcaster decides to change broadcast channel or date\&time, the ScheduledRecording service is supposed to track.

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

Request:
CreateRecordSchedule(""
<?xml version="1.0" encoding="UTF-8"?>
<srs
xmlns="urn:schemas-upnp-org:av:srs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
  urn:schemas-upnp-org:av:srs
  http://www.upnp.org/schemas/av/srs.xsd">
  <item id="">
    <title>Meet the UPnP Guys series</title>
    <class>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</class>
    <desiredPriority type="PREDEF">L1</desiredPriority>
    <recordDestination mediaType="HDD" preference="1">
      Hard Disk
    </recordDestination>
    <desiredRecordQuality type="DEFAULT">
      SD, ED
    </desiredRecordQuality>
    <matchingName type="SERIES" subStringMatch="0">
      Meet the UPnP Guys
    </matchingName>
    <matchingStartDateTimeRange>
      NOW/09-30T23:59:59
    </matchingStartDateTimeRange>
    <totalDesiredRecordTasks>0</totalDesiredRecordTasks>
    <channelMigration>1</channelMigration>
    <timeMigration>1</timeMigration>
  </item>
</srs>"")

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

Response:
CreateRecordSchedule("s201","
<?xml version="1.0" encoding="UTF-8"?>
<srs
xmlns="urn:schemas-upnp-org:av:srs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
  urn:schemas-upnp-org:av:srs
  http://www.upnp.org/schemas/av/srs.xsd">
  <item id="s201">
    <title>Meet the UPnP Guys series</title>
    <class>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</class>
    <priority>L1</priority>
    <desiredPriority type="PREDEF">L1</desiredPriority>
    <recordDestination mediaType="HDD" preference="1">
      Hard Disk
    </recordDestination>
  </item>
</srs>"
2.9.3.2.2 Creating a `object.recordSchedule.query.contentID` Class `recordSchedule`

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

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

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

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

The recordings must be stored on the internal Hard Disk. If, for some reason, the Hard Disk is not available at the time of recording, the recording must be canceled. The control point also specifies that the recording...
should be encoded using a high record quality setting of “720p60”. If that is not possible, no recording will be made. No pre-roll or post-roll times are specified. If the broadcaster decides to change broadcast channel or date&time, the ScheduledRecording service is supposed to track.

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

**Request:**
CreateRecordSchedule(""
<?xml version="1.0" encoding="UTF-8"?>
<srs
 xmlns="urn:schemas-upnp-org:av:srs"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
 urn:schemas-upnp-org:av:srs
 http://www.upnp.org/schemas/av/srs.xsd">
 <item id="">
 <title>My Program</title>
 <class>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</class>
 <desiredPriority type="PREDEF">L1</desiredPriority>
 <recordDestination mediaType="HDD" preference="1">Hard Disk</recordDestination>
 <desiredRecordQuality type="ATSC">720p60</desiredRecordQuality>
 <matchingID type="MyLocalProvider.net">123456</matchingID>
 <totalDesiredRecordTasks>1</totalDesiredRecordTasks>
 <channelMigration>1</channelMigration>
 <timeMigration>1</timeMigration>
 </item>
 </srs>"

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

**Response:**
CreateRecordSchedule("s202",""
<?xml version="1.0" encoding="UTF-8"?>
<srs
 xmlns="urn:schemas-upnp-org:av:srs"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
 urn:schemas-upnp-org:av:srs
 http://www.upnp.org/schemas/av/srs.xsd">
 <item id="s202">
 <title>My Program</title>
 <class>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</class>
 <priority>L1</priority>
 <desiredPriority type="PREDEF">L1</desiredPriority>
 <recordDestination mediaType="HDD" preference="1">Hard Disk</recordDestination>
 <desiredRecordQuality type="ATSC">720p60</desiredRecordQuality>
 <matchingID type="MyLocalProvider.net">123456</matchingID>
 <totalDesiredRecordTasks>1</totalDesiredRecordTasks>
 <channelMigration>1</channelMigration>
 <timeMigration>1</timeMigration>
 </item>
 </srs>"
2.9.4 Deleting a recordSchedule

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

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

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

Request:
DeleteRecordSchedule("s301")

Response:
---

2.9.5 Browsing recordSchedule and recordTask instances

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

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

recordSchedule (@id = “s101”, class = “OBJECT.RECORDSCHEDULE.DIRECT.MANUAL”)
recordTask (@id = “t101-001”, class = “OBJECT.RECORDTASK”)
recordTask (@id = “t101-002”, class = “OBJECT.RECORDTASK”)

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
When a control point wants to gather detailed information on currently existing recordSchedule instances, it can do this by invoking the BrowseRecordSchedules() action. The following request:

**Request:**
BrowseRecordSchedules("", 0, 10, "+srs:title")

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

**Response:**
BrowseRecordSchedules(""
  <title>About SRS</title>
  <class>OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE</class>
  <priority>L1</priority>
  <recordDestination mediaType="DVD+RW" preference="1">
    DVD Drive
  </recordDestination>
  <recordDestination mediaType="HDD" preference="2">
    Hard Disk
  </recordDestination>
  <scheduledProgramCode type="upnpexample.org_upnpProgramCode">
    2005-07-01_09:00:00_00:30:00
  </scheduledProgramCode>
  <scheduleState currentErrors="">
    OPERATIONAL
  </scheduleState>
  <abnormalTasksExist>0</abnormalTasksExist>
  <currentRecordTaskCount>1</currentRecordTaskCount>
</item><item id="s101">
  <title>BBC News at 7pm</title>
</item></srs>
<class>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</class>
<priority>L2</priority>
<recordDestination mediaType="HDD" preference="1">
  Hard Disk
</recordDestination>
<recordDestination mediaType="DVD+R" preference="2">
  DVD Recorder
</recordDestination>
<scheduledChannelID type="ANALOG">47</scheduledChannelID>
<scheduledStartDateTime>T19:00:00</scheduledStartDateTime>
<scheduledDuration>P01:00:00</scheduledDuration>
<scheduleState
  currentErrors="">
  OPERATIONAL
</scheduleState>
<abnormalTasksExist>0</abnormalTasksExist>
<currentRecordTaskCount>2</currentRecordTaskCount>

</item>

<item id="s103">
<title>Life of a Software Developer</title>
<class>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</class>
<priority>L1</priority>
<recordDestination mediaType="HDD" preference="1">
  Hard Disk
</recordDestination>
<recordDestination mediaType="HDD" preference="2">
  targetURL="http://192.168.0.12/MyNAS/RecordedTV">
  Shared Content
</recordDestination>
<scheduledCDSObjectID>
  User_Channel_5
</scheduledCDSObjectID>
<scheduledStartDateDateTime>
  MONT19:00:00
</scheduledStartDateDateTime>
<scheduledDuration>P01:00:00</scheduledDuration>
<scheduleState
  currentErrors="">
  OPERATIONAL
</scheduleState>
<abnormalTasksExist>0</abnormalTasksExist>
<currentRecordTaskCount>2</currentRecordTaskCount>

</item>

<item id="s201">
<title>Meet the UPnP Guys series</title>
<class>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</class>
<priority>L1</priority>
<recordDestination mediaType="HDD" preference="1">
  Hard Disk
</recordDestination>
<matchingName type="SERIES" subStringMatch="0">
  Meet the UPnP Guys
</matchingName>
<scheduleState
  currentErrors="">
  OPERATIONAL
</scheduleState>
<abnormalTasksExist>0</abnormalTasksExist>
<currentRecordTaskCount>1</currentRecordTaskCount>

</item>
2.9.5.2 Browsing recordTask instances associated with a single recordSchedule

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

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

The following request:

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

returns the following response:

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

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
urn:schemas-upnp-org:av:srs
http://www.upnp.org/schemas/av/srs.xsd">
  <item id="t101-001">
    <title>BBC News at 7pm</title>
    <class>OBJECT.RECORDTASK</class>
    <cdsReference link="LINK1">
      <!-- The following DIDL-Lite XML Document needs to be interpreted as a simple string and therefore needs to be properly escaped -->
      &lt;?xml version="1.0" encoding="UTF-8"?&gt;
      &lt;DIDL-Lite
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
        xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
        http://www.upnp.org/schemas/av/didl-lite.xsd
        urn:schemas-upnp-org:metadata-1-0/upnp/
        http://www.upnp.org/schemas/av/upnp.xsd"
        &lt;item id="18" parentID="13" restricted="0"&gt;
          &lt;dc:title&gt;BBC News at 7pm&lt;/dc:title&gt;
          &lt;upnp:dc:title&gt;BBC News at 7pm&lt;/upnp:dc:title&gt;
          &lt;upnp:class&gt;
          object.item.videoItem.videoBroadcast
          &lt;/upnp:class&gt;
          &lt;dc:creator&gt;BBC&lt;/dc:creator&gt;
        &lt;/item&gt;
        &lt;/DIDL-Lite&gt;
    </cdsReference>
    <priority>L2</priority>
    <desiredPriority type="PREDEF">L2</desiredPriority>
    <recordDestination mediaType="HDD" preference="1">
      Hard Disk
    </recordDestination>
    <recordDestination mediaType="DVD+R" preference="2">
      DVD Recorder
    </recordDestination>
    <desiredRecordQuality type="DEFAULT">
      SD,AUTO
    </desiredRecordQuality>
    <recordQuality type="DEFAULT">
      SD
    </recordQuality>
    <recordQuality type="ATSC">
      480i60
    </recordQuality>
    <recordQuality type="QLEVEL">
      Q3
    </recordQuality>
    <recordScheduleID>s101</recordScheduleID>
    <recordedCDSObjectID link="LINK1">
      rec00001
    </recordedCDSObjectID>
    <taskChannelID type="ANALOG">47</taskChannelID>
  </item>
</ScheduledRecording:2 Service Template Version 1.01 – Document Version 1.0
<taskStartDateTime>2005-06-29T19:00:00</taskStartDateTime>
<taskDuration>P01:00:00</taskDuration>
<taskStartDateTimeAdjust>-P00:02:30</taskStartDateTimeAdjust>
<taskDurationAdjust>+P00:05:00</taskDurationAdjust>
<taskState phase="IDLE"
startDateTimeMet="0"
endDateTimeMet="0"
recording="0"
someBitsRecorded="0"
someBitsMissing="0"
firstBitsRecorded="0"
lastBitsRecorded="0"
fatalError="0"
currentErrors=""
errorHistory=""
pendingErrors=""
infoList="">
  IDLE.READY
</taskState>
</item>
</item id="t101-002">
<title>BBC News at 7pm</title>
class=OBJECT.RECORDTASK</class>
<cdsReference link="LINK1">
  <!-- The following DIDL-Lite XML Document needs to be interpreted as a simple string and therefore needs to be properly escaped -->
  <![CDATA[
<?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"&gt;
  &lt;item id="18" parentID="13" restricted="0"&gt;
    &lt;dc:title&gt;BBC News at 7pm&lt;/dc:title&gt;
    &lt;dc:creator&gt;BBC&lt;/dc:creator&gt;
    &lt;upnp:class&gt;
      object.item.videoItem.videoBroadcast
    &lt;/upnp:class&gt;
  &lt;/item&gt;
&lt;/didl-lite&gt;]]>
</cdsReference>
2.9.6 Rating System

A ScheduledRecording service offers the OPTIONAL ability to impose rating limits on recordable content.
A ScheduledRecording service implementation may provide a list of supported ratings. The supported ratings can be retrieved by invoking the GetAllowedValues() action and specifying the matchingRatingLimit property in the Filter argument.

In the United States, TV manufacturers are REQUIRED to provide built-in support for the TV Parental Guidelines Monitoring Board rating system. (See http://www.tvguidelines.org.) Motion picture content is rated on a voluntary basis by the Motion Picture Association of America. (See http://www.mpaa.org.) Since it is not a simple matter to determine the rating system applicable to recordable content, the control point should provide values for all applicable rating systems when specifying a rating limit.

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

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

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

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

### 2.9.7 Conflict Detection and Resolution

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

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

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

Alternatively, a control point may use the DeleteRecordSchedule() and CreateRecordSchedule() actions to reschedule the recordSchedule with a different priority level. The drawback of this approach is that all recordTask instances associated with the deleted recordSchedule are also deleted and any customization by the user that happened at the recordTask level will get lost as well.
A future version of this specification will address the conflict resolution issue in detail.
3 XML Service Description

```xml
<?xml version="1.0" encoding="UTF-8"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>GetSortCapabilities</name>
      <argumentList>
        <argument>
          <name>SortCaps</name>
          <direction>out</direction>
          <relatedStateVariable>
            SortCapabilities
          </relatedStateVariable>
        </argument>
        <argument>
          <name>SortLevelCap</name>
          <direction>out</direction>
          <relatedStateVariable>
            SortLevelCapability
          </relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetPropertyList</name>
      <argumentList>
        <argument>
          <name>DataTypeID</name>
          <direction>in</direction>
          <relatedStateVariable>
            A_ARG_TYPE_DataTypeID
          </relatedStateVariable>
        </argument>
        <argument>
          <name>PropertyList</name>
          <direction>out</direction>
          <relatedStateVariable>
            A_ARG_TYPE_PropertyList
          </relatedStateVariable>
        </argument>
      </argumentList>
    </action>
    <action>
      <name>GetAllowedValues</name>
      <argumentList>
        <argument>
          <name>DataTypeID</name>
          <direction>in</direction>
          <relatedStateVariable>
            A_ARG_TYPE_DataTypeID
          </relatedStateVariable>
        </argument>
      </argumentList>
    </action>
  </actionList>
</scpd>
```
<action>
   <name>GetStateUpdateID</name>
   <argumentList>
      <argument>
         <name>Id</name>
         <direction>out</direction>
         <relatedStateVariable>
            StateUpdateID
         </relatedStateVariable>
      </argument>
   </argumentList>
</action>

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

<argument>
  <name>Result</name>
  <direction>out</direction>
  <relatedStateVariable>
    A_ARG_TYPE_RecordSchedule
  </relatedStateVariable>
</argument>

<argument>
  <name>NumberReturned</name>
  <direction>out</direction>
  <relatedStateVariable>
    A_ARG_TYPE_Count
  </relatedStateVariable>
</argument>

<argument>
  <name>TotalMatches</name>
  <direction>out</direction>
  <relatedStateVariable>
    A_ARG_TYPE_Count
  </relatedStateVariable>
</argument>

<argument>
  <name>UpdateID</name>
  <direction>out</direction>
  <relatedStateVariable>
    StateUpdateID
  </relatedStateVariable>
</argument>

</argumentList>
</action>

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

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

</action>

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

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

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

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

<action>
  <name>Result</name>
  <direction>out</direction>
  <relatedStateVariable>
    A_ARG_TYPE_RecordSchedule
  </relatedStateVariable>
</action>

<action>
  <name>UpdateID</name>
  <direction>out</direction>
  <relatedStateVariable>
    StateUpdateID
  </relatedStateVariable>
</action>

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

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

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

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

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

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

<argument>
  <name>RecordTaskConflictIDList</name>
  <direction>out</direction>
  <relatedStateVariable>
    A_ARG_TYPE_ObjectIDList
  </relatedStateVariable>
</argument>

<argument>
  <name>UpdateID</name>
  <direction>out</direction>
  <relatedStateVariable>
    StateUpdateID
  </relatedStateVariable>
</argument>

</argumentList>
</action>

</actionList>

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

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

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

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

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

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

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
4 Test
No semantic tests have been specified for this service.
Appendix A.  srs XML Document (Normative)

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

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

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

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

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

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

A.1 A_ARG_TYPE_RecordSchedule AVDT XML Document

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

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

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

Field specific rules:
• There must be one and only one field with the subelement <name> set to “class”
• The allowed values for this field MUST only be derived from the object.recordSchedule virtual class.
• The <name> subelement of all <field> elements MUST only contain names of recordSchedule properties.

For a full-fledged example of a A_ARG_TYPE_RecordSchedule AVDT XML Document, see Appendix G.1, “A ARG TYPE_RecordSchedule AVDT Example”

A.2 A_ARG_TYPE_RecordTask AVDT XML Document

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

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

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

Field specific rules:

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

For a full-fledged example of a A_ARG_TYPE_RecordTask AVDT XML Document, see Appendix G.2, “A ARG TYPE_RecordTask AVDT Example”.

A.3 A_ARG_TYPE_RecordScheduleParts AVDT XML Document

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

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

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

Field specific rules:

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

For a full-fledged example of a `A_ARG_TYPE_RecordScheduleParts AVDT XML Document`, see Appendix G.3, “A_ARG_TYPE_RecordScheduleParts AVDT Example”.


Appendix B. AV Working Committee Extended Properties (Normative)

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

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

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

B.1 Base Properties

Table B-1: Base Properties Overview

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>@id</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.1.1</td>
</tr>
<tr>
<td>title</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.1.2</td>
</tr>
<tr>
<td>class</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.1.3</td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.1.4</td>
</tr>
<tr>
<td>cdsReference</td>
<td>srs</td>
<td>xsd:string</td>
<td>YES</td>
<td>Appendix B.1.5</td>
</tr>
<tr>
<td>cdsReference@link</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.1.5.1</td>
</tr>
</tbody>
</table>

B.1.1  @id

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

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

Default Value: N/A – Required on input.

Sort Order: Lexical or Lexical Numeric.

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

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

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

B.1.2  title

Namespace: srs  Property Data Type: xsd:string  Multi-Valued: NO
Description: Primary title of the object. The title property contains a friendly name to identify the object. This property can be either user-supplied or derived from the content name the object represents. This property is not to be confused with the matchingName or matchedName property. See also http://dublincore.org/documents/dces.

Default Value: N/A – Required on input.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

B.1.3 class

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

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

Default Value: N/A – Required on input.

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

Input: The desired setting.

Output: The current setting.

B.1.3.1 allowedValueList for the class Property

Table B-2: allowedValueList for the class Property

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“OBJECT.RECORDSCHEDULE.DIRECT.MANUAL”</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>“OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG”</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>“OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG”</td>
<td>R</td>
<td>Control points should support all predefined values in these rows.</td>
</tr>
<tr>
<td>“OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE”</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>“OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME”</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>“OBJECT.RECORDSCHEDULE.QUERY.CONTENTID”</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>“OBJECT.RECORDTASK”</td>
<td>Q</td>
<td></td>
</tr>
</tbody>
</table>

vendor-defined.

Vendor-defined class names MUST obey the rules set forth in Appendix D.3, “Class Name Syntax”.

B.1.4 additionalStatusInfo

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

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

Default Value: N/A – Output only.
Sort Order: Lexical.
Input: N/A.
Output: The current setting.

B.1.5  **cdsReference**

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

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

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

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

Default Value: N/A – Output only.
Sort Order: Sorting on this property is meaningless and will be ignored.
Input: N/A.
Output: The current setting.

B.1.5.1  **cdsReference@link**

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

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

Default Value: N/A – Output only.
Sort Order: Lexical or Lexical Numeric.

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

Input: N/A.
Output: The current setting.

**B.2 Priority Properties**

**Table B-3: Priority Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.2.1</td>
</tr>
</tbody>
</table>
### B.2.1 priority

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

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

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

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

**Default Value:** N/A – Output only.

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

**Input:** N/A.

**Output:** The current setting.

#### B.2.1.1 allowedValueList for the `priority` Property

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

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“LI”</td>
<td>R</td>
<td>The highest priority level supported by the device.</td>
</tr>
<tr>
<td>“L2”</td>
<td>O</td>
<td>The next progressively lower priority level supported by the device.</td>
</tr>
<tr>
<td>…</td>
<td>O</td>
<td>Progressively lower priority level supported by the device.</td>
</tr>
<tr>
<td>“L&lt;x&gt;”</td>
<td>O</td>
<td>The lowest priority level supported by the device where &lt;x&gt; is the total number of distinct priority levels supported by the device.</td>
</tr>
</tbody>
</table>

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

If “L<x>” is supported, then all values between “LI” and “L<x>” MUST be supported.

#### B.2.1.2 priority@orderedValue

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

**Description:** The `priority@orderedValue` property indicates the relative numerical priority value of the associated object (a `recordSchedule` or a `recordTask`). A value of 1 indicates that this object is the highest priority object of that object type (that is: of all `recordSchedule` instances or all `recordTask` instances). Other ascending values indicate that the object has a progressively lower priority relative to the other objects of that type. A value of N (where N is the total number of objects of that type) indicates that the
object is the lowest priority object of that type. No two objects of the same type will have the same value for this property.

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

Default Value: N/A – Output only.

Sort Order: Numeric.

Input: N/A.

Output: The current setting.

B.2.2 desiredPriority

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

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

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

Default Value: “DEFAULT”.

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

Input: The desired setting.

Output: The current setting.

B.2.2.1 allowedValueLists for the desiredPriority Property

Table B-5: Primary allowedValueList for the desiredPriority Property

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“DEFAULT”</td>
<td>R</td>
<td>No priority preference. The device itself will determine the object’s priority.</td>
</tr>
<tr>
<td>“L1”</td>
<td>O</td>
<td>The highest priority level supported by the device.</td>
</tr>
<tr>
<td>“L2”</td>
<td>O</td>
<td>The next to highest priority level supported by the device.</td>
</tr>
<tr>
<td>...</td>
<td>O</td>
<td>Progressively lower priority levels between 1 and &lt;x&gt;, “L3”, “L4”, etc.</td>
</tr>
<tr>
<td>“L&lt;x&gt;”</td>
<td>O</td>
<td>The lowest priority level supported by the device where &lt;x&gt; is the total number of distinct priority levels supported by the device.</td>
</tr>
</tbody>
</table>

Notes:

All devices MUST support 1 or more priority levels.

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

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
Additionally, if the device supports the `priority@orderedValue` property, then the device MUST also support the following allowed values. Conversely, if any of these allowed values are supported, then the device MUST support the `priority@orderedValue` property. These allowed values provide a mechanism for more precise prioritization control with those devices that support it.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;HIGHEST&quot;</td>
<td>R</td>
<td>The highest level possible. – Same as “L1 HI” defined below.</td>
</tr>
<tr>
<td>&quot;LOWEST&quot;</td>
<td>R</td>
<td>The lowest level possible. – Same as “L&lt;x&gt; LOW” defined below.</td>
</tr>
<tr>
<td>&quot;L1 HI&quot;</td>
<td>R</td>
<td>The highest priority possible within the highest priority level.</td>
</tr>
<tr>
<td>&quot;L1 LOW&quot;</td>
<td>R</td>
<td>The lowest priority possible within the highest priority level.</td>
</tr>
<tr>
<td>&quot;L2 HI&quot;</td>
<td>R</td>
<td>The highest priority possible within the next to highest priority level.</td>
</tr>
<tr>
<td>&quot;L2 LOW&quot;</td>
<td>R</td>
<td>The lowest priority possible within the next to highest priority level.</td>
</tr>
<tr>
<td>...</td>
<td>R</td>
<td>Progressively lower priority levels.</td>
</tr>
<tr>
<td>&quot;L&lt;x&gt; HI&quot;</td>
<td>R</td>
<td>The highest priority possible within the lowest priority level where &lt;x&gt; is the total number of distinct priority levels supported by the device.</td>
</tr>
<tr>
<td>&quot;L&lt;x&gt; LOW”</td>
<td>R</td>
<td>The lowest priority possible, but within the lowest priority level where &lt;x&gt; is the total number of distinct priority levels supported by the device.</td>
</tr>
<tr>
<td>&lt;@id&gt;</td>
<td>R</td>
<td>The next highest priority “slot” immediately higher than (but within the same priority level of) the existing object whose @id is specified by &lt;@id&gt;.</td>
</tr>
</tbody>
</table>

Notes:

1. If a device supports the `priority@orderedValue` property, then the device MUST also support these CONDITIONALLY REQUIRED allowed values. Conversely, if any of these allowed values are supported, then the device MUST support the `priority@orderedValue` property.
2. These allowed values provide a mechanism for more precise prioritization control with those devices that support it. If “L<x> LOW” is supported, then all values between “L1 HI” and “L<x> LOW” MUST be continuously supported; that is: a device MUST not support only “L1 HI”, “L1 LOW”, “L3 HI”, “L3 LOW”, “L5 HI” and “L5 LOW” or only “L1 HI”, “L2 HI”, and “L3 HI”.

B.2.3 `desiredPriority@type` Property

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

Description: When the `desiredPriority@type` property is set to “PREDEF”, it indicates that the `desiredPriority` property contains one of the predefined priority labels (“L1”, “L2 LOW”, etc.). When set to “OBJECTID”, it indicates that the `desiredPriority` property contains an object ID (@id value).

Default Value: “PREDEF”.

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
Sort Order: Sorting on this property is meaningless and will be ignored.

Input: The desired setting.

Output: The current setting.

B.2.3.1 allowedValueLists for the desiredPriority@type Property

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

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;PREDEF&quot;</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>&quot;OBJECTID&quot;</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>

B.3 Output Control Properties

Table B-8: Output Control Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordDestination</td>
<td>srs</td>
<td>xsd:string</td>
<td>YES</td>
<td>Appendix B.3.1</td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td>srs</td>
<td>CSV (xsd:string)</td>
<td>NO</td>
<td>Appendix B.3.1.1</td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td>srs</td>
<td>xsd:anyURI</td>
<td>NO</td>
<td>Appendix B.3.1.2</td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td>srs</td>
<td>xsd:unsignedInt</td>
<td>NO</td>
<td>Appendix B.3.1.3</td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.3.2</td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.3.2.2</td>
</tr>
</tbody>
</table>

B.3.1 recordDestination

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

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

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

Examples: “Hard Disk Drive”, “DVD-1”, “LINE1”, “AUX-out” etc.

Default Value: Vendor-defined.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

B.3.1.1 recordDestination@mediaType

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

Description: The recordDestination@mediaType property indicates the type of media that is to be used for the recording.
If the media type of the specified record destination is fixed (not removable), then the value of the recordDestination@mediaType property MUST match the actual physical media type of the record destination. This single value can be retrieved through the GetAllowedValues() action.

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

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

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

Examples: “HDD”, “DVD-RW”

Default Value: Vendor-defined.

Sort Order: Sequenced Lexical.

Input: The desired setting.

Output: The current setting.

B.3.1.1 allowedValueList for the recordDestination@mediaType Property

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

B.3.1.2 recordDestination@targetURL

Namespace: srs

Property Data Type: xsd:anyURI

Multi-Valued: NO

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

Examples:

“file:///D:/MyDocuments/MyVideos”

“http://10.0.0.1/MyDocuments/MyVideos”

Default Value: Vendor-defined.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

B.3.1.3 recordDestination@preference

Namespace: srs

Property Data Type: xsd:unsignedInt

Multi-Valued: NO

Description: The recordDestination@preference property is useful when multiple recordDestination properties are specified within the same recordSchedule or recordTask object. In this case, the values...
indicate the preference order of the multiple record destinations. Higher numbers indicate lower preference. The values do not have to be contiguous.

If multiple `recordDestination@preference` properties have the same value, then the order of preference in which their associated record destinations are chosen is device-dependent.

If the `recordDestination@preference` property is not supported by an implementation, then the order of preference of all specified record destinations is device-dependent.

**Default Value:** Vendor-defined.

**Sort Order:** Numeric.

**Input:** The desired setting.

**Output:** The current setting.

### B.3.2 `desiredRecordQuality`

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

**Description:** The `desiredRecordQuality` property is used to express the desired or preferred recording quality level(s) for a particular `recordSchedule`. Multiple recording quality levels can be specified in the comma-separated value list of the `desiredRecordQuality` property. If there is more than one value specified, then the values indicate the desired recording quality, in order of preference, highest preference first. The value “AUTO” MUST be supported by all implementations. When “AUTO” is included in the list, it MUST appear as the last value in the list and indicates that if none of the preceding values are available, then the ScheduledRecording service is free to use any recording quality level to maximize the probability that the recording actually takes place. When the “AUTO” value is the only value in the list, then the ScheduledRecording service is free to use any recording quality level.

There are many ways to express recording quality. Some implementations use bitrates, some use user-friendly labels etc. Some implementations might even support multiple ways to express recording quality simultaneously. The `desiredRecordQuality` property is used in conjunction with the `desiredRecordQuality@type` to allow implementations to express these variations. However, since the `desiredRecordQuality` property can appear only once, the acceptable recording quality levels for a particular `recordSchedule` are restricted to a single type variation.

If an implementation is capable of encoding or transcoding, then it MAY do so in order to achieve the desired recording quality.

**Example:** Assume a (hypothetical) implementation that supports the type variations “DEFAULT”, “ATSC” and “QLEVEL” for the `desiredRecordQuality@type` property. The following table expresses the supported `desiredRecordQuality` property values for those variations and also indicates how the different type variations interrelate for this particular implementation:
Table B-9:  \texttt{desiredRecordQuality} Example

<table>
<thead>
<tr>
<th>“\texttt{DEFAULT}”</th>
<th>“\texttt{ATSC}”</th>
<th>“\texttt{QLEVEL}”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“\texttt{HD}”</td>
<td>“1080p30”</td>
<td>“Q1”</td>
</tr>
<tr>
<td></td>
<td>“1080p24”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“1080i60”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“720p60”</td>
<td>“Q2”</td>
</tr>
<tr>
<td></td>
<td>“720p30”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“720p24”</td>
<td></td>
</tr>
<tr>
<td>“\texttt{ED}”</td>
<td>“480p60”</td>
<td></td>
</tr>
<tr>
<td>“\texttt{SD}”</td>
<td>“480p30”</td>
<td>“Q3”</td>
</tr>
<tr>
<td></td>
<td>“480p24”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“480i60”</td>
<td></td>
</tr>
<tr>
<td>“\texttt{AUTO}”</td>
<td>“AUTO”</td>
<td>“AUTO”</td>
</tr>
</tbody>
</table>

- Specifying “\texttt{HD,ED}” in the \texttt{desiredRecordQuality} property and “\texttt{DEFAULT}” in the \texttt{desiredRecordQuality@type} property will result in the following:
  - If possible, the recording will be made using “\texttt{HD}” quality. In this case, it is up to the implementation to determine exactly which recording quality level within the “\texttt{HD}” range will be used for the recording.
  - If recording using “\texttt{HD}” quality is not possible, the recording will be made using “\texttt{ED}” quality, if possible. Again, it is up to the implementation to determine exactly which recording quality level within the “\texttt{ED}” range will be used for the recording.
  - If the recording cannot be made in either “\texttt{HD}” or “\texttt{ED}” quality, then no recording will be made.

- Specifying “\texttt{ED,SD,AUTO}” in the \texttt{desiredRecordQuality} property and “\texttt{DEFAULT}” in the \texttt{desiredRecordQuality@type} property will result in the following:
  - If possible, the recording will be made using “\texttt{ED}” quality. It is up to the implementation to determine exactly which recording quality level within the “\texttt{ED}” range will be used for the recording.
  - If that is not possible, the recording will be made using “\texttt{SD}” quality, if possible. It is up to the implementation to determine exactly which recording quality level within the “\texttt{SD}” range will be used for the recording.
  - If the recording cannot be made in either “\texttt{ED}” or “\texttt{SD}” quality, then the recording will be made using any other available recording quality.

- Specifying “\texttt{720p60}” in the \texttt{desiredRecordQuality} property and “\texttt{ATSC}” in the \texttt{desiredRecordQuality@type} property will result in the following:
  - If possible, the recording will be made using “\texttt{720p60}” quality.
  - If that is not possible, no recording will be made.

When the ScheduledRecording service responds to a \texttt{GetAllowedValues()} action with \texttt{desiredRecordQuality} information, then the allowed values MUST be listed in order of quality from highest quality to lowest. The value “\texttt{AUTO}” MUST always be present and appear as the last item in the list.

Copyright © 2008, Contributing Members of the UPnPTM Forum. All rights Reserved.
Default Value: Vendor-defined.

Sort Order: Property Specific, based on the allowedValueList for the desiredRecordQuality property. Ascending: lowest quality first.

Input: The desired setting.

Output: The current setting.

B.3.2.1 allowedValueList for the desiredRecordQuality Property

Table B-10: allowedValueList for the desiredRecordQuality Property

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“AUTO”</td>
<td>R</td>
<td>If none of the quality levels preceding the “AUTO” value are available, then any recording quality level may be used. The “AUTO” value MUST always appear last in the list when present.</td>
</tr>
<tr>
<td>Vendor-defined</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

B.3.2.2 desiredRecordQuality@type

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

Description: There are many ways to express recording quality. Some implementations use bitrates, some use user-friendly labels etc. Some implementations might even support multiple ways to express recording quality simultaneously. The desiredRecordQuality@type property is used to express which type variation is used in its associated independent desiredRecordQuality property. The “DEFAULT” value MUST be supported and indicates which of the supported type variations is preferred by the device when expressing recording quality levels.

Default Value: Vendor-defined.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

B.3.2.2.1 allowedValueList for the desiredRecordQuality@type Property

Table B-11: allowedValueList for the desiredRecordQuality@type Property

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“DEFAULT”</td>
<td>R</td>
<td>Indicates the type variation that is preferred by the device when expressing recording quality levels.</td>
</tr>
<tr>
<td>Vendor-defined</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
B.4  Content Identification Related Properties

Table B-12:  Content Identification Related Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduledCDSObjectID</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.4.1</td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.4.1.1</td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.4.2</td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.4.2.1</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkName</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.4.2.2</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkID</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.4.2.3</td>
</tr>
<tr>
<td>scheduledStartDateTime</td>
<td>srs</td>
<td>xsd:string</td>
<td>YES</td>
<td>Appendix B.4.3</td>
</tr>
<tr>
<td>scheduledDuration</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.4.4</td>
</tr>
<tr>
<td>scheduledProgramCode</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.4.5</td>
</tr>
<tr>
<td>scheduledProgramCode@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.4.5.1</td>
</tr>
</tbody>
</table>

B.4.1  scheduledCDSObjectID

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

Description: The scheduledCDSObjectID property contains the didl-lite:@id property value of the ContentDirectory service object from which relevant metadata information is extracted to create the recordSchedule.

Default Value: N/A – Required on input.

Sort Order: Lexical or Lexical Numeric.

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

Input: The desired setting.

Output: The current setting.

B.4.1.1  scheduledCDSObjectID@link

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

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

Default Value: N/A – Output only.

Sort Order: Same as cdsReference@link.

Input: N/A.

Output: The current setting.
B.4.2  **scheduledChannelID**

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

**Description**: The `scheduledChannelID` property provides channel information for the `recordSchedule`. Its format depends on the `scheduledChannelID@type` property as follows:

If `scheduledChannelID@type` = “ANALOG” then the `scheduledChannelID` property contains the channel number.

**Note**: The label “ANALOG” actually is a misnomer. The format of the `scheduledChannelID` property does not indicate the nature of the channel (analog versus digital). Rather, it merely indicates the format of the label (for example, a single channel number versus a channel number pair). Therefore, the label “ANALOG” should be interpreted as meaning: the `scheduledChannelID` property’s format is a single channel number.

Examples: “5”, “7”, etc.

If `scheduledChannelID@type` = “DIGITAL” then the `scheduledChannelID` property contains the channel number pair “<Major Channel Number>,<Minor Channel Number>”.

**Note**: The label “DIGITAL” actually is a misnomer. The format of the `scheduledChannelID` property does not indicate the nature of the channel (analog versus digital). Rather, it merely indicates the format of the label (for example, a single channel number versus a channel number pair). Therefore, the label “DIGITAL” should be interpreted as meaning: the `scheduledChannelID` property’s format is a channel number pair.

Examples: “5,1”, “5,2”, etc.

If `scheduledChannelID@type` = “FREQUENCY” then the `scheduledChannelID` property contains the channel center frequency, expressed in Hz.

Examples: “150125000” (VHF band), “615000000” (UHF band), “965000000” (FM band), etc.

If `scheduledChannelID@type` = “SI” then the `scheduledChannelID` property contains the Service Information Triplet “<Network ID>,<Transport Stream ID>, <Service ID>” , embedded in the content stream.

Examples: “0x1234,0xFEDC,0x0102”, “12345,23456,32109”, etc.

If `scheduledChannelID@type` = “LINE” then the `scheduledChannelID` property contains a vendor-defined label identifying the line input.

Examples: “Line 1”, “AUX”, “Front”, “Rear”, etc.

If `scheduledChannelID@type` = “NETWORK” then the `scheduledChannelID` property contains the URI that uniquely identifies the content to be recorded.


**Default Value**: N/A – Required on input.

**Sort Order**: type Relationship.

“ANALOG”: Numeric.

“DIGITAL”: Sequenced numeric.

“FREQUENCY”: Numeric.

“SI”: Sequenced lexical.

“LINE”: Lexical.

“NETWORK”: Lexical.
Vendor-defined: Vendor-defined sorting.

Input: The desired setting.

Output: The current setting.

B.4.2.1  **scheduledChannelID@type**

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

Description: The *scheduledChannelID@type* property determines the format that is used for the *scheduledChannelID* property as defined above.

Default Value: N/A – Required on input.

Sort Order: Property Specific, based on the order in Table B-13. Ascending: first table entry first. If there is a single vendor-defined value, it sorts in table position. If there are multiple vendor-defined values, they sort lexically among themselves, all after the Table B-13 entries in ascending order and all before the Table B-13 entries in descending order.

Input: The desired setting.

Output: The current setting.

**Table B-13: allowedValueList for the **scheduledChannelID@type** Property**

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ANALOG&quot;</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>&quot;DIGITAL&quot;</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>&quot;FREQUENCY&quot;</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>&quot;ST&quot;</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>&quot;LINE&quot;</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>&quot;NETWORK&quot;</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>Vendor-defined</td>
<td>X</td>
<td>At least one value in these rows MUST be supported by a compliant ScheduledRecording service implementation. Control points should support all values in these rows.</td>
</tr>
</tbody>
</table>

B.4.2.2  **scheduledChannelID@distriNetworkName**

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

Description: The *scheduledChannelID@distriNetworkName* property contains the name of the Distribution Network that provides the channel, identified by the *scheduledChannelID* property.

Default Value: Vendor-defined.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

B.4.2.3  **scheduledChannelID@distriNetworkID**

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

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
Description: The scheduledChannelID@districtNetworkID property contains an ID that uniquely identifies the Distribution Network that provides the channel, identified by the scheduledChannelID property.

Default Value: Vendor-defined.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

B.4.3 scheduledStartDateTime

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

Description: The scheduledStartDateTime property indicates what date or day(s) and time the recording will take place. This property does not account for any recording time adjustments such as scheduledStartDateTimeAdjust, and device latencies.

The sched-start format syntax of the scheduledStartDateTime property is defined in Appendix D, “EBNF Syntax Definitions”.

Examples: “02-07T15:30:00” (February 7th, 3:30pm), “2005-02-07T15:30:00” (February 7th, 2005, 3:30pm), “MONT15:30:00” (Mondays at 3:30pm), “T15:30:00” (Every day at 3:30pm)

Recording(s) will occur on the next occurrence(s) of the specified date or day(s) and time until the total number of desired recordings (as indicated by the totalDesiredRecordTasks property) has been made.

Note that the scheduledStartDateTime property is a multi-valued property. Therefore, multiple date&times can be specified for the same recordSchedule. Recording will occur on every next occurrence of any of the specified start date&times until the total number of desired recordings (as indicated by the totalDesiredRecordTasks property) has been made.

See Appendix B.7.1, “totalDesiredRecordTasks” for further details on the use of the totalDesiredRecordTasks property.

The value “NOW” is defined by this specification to indicate that the recording MUST start immediately (as soon as possible).

Default Value: N/A – Required on input.

Sort Order: Property Specific, in chronological order.

Input: The desired setting.

Output: The current setting.

B.4.4 scheduledDuration

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

Description: The scheduledDuration property indicates the scheduled duration of the recording. The duration format syntax of the scheduledDuration property is defined in Appendix D, “EBNF Syntax Definitions”.

Examples: “P01:30:00” (one hour and thirty minutes), “P2D01:15:00” (two-day and seventy five minutes recording).

This property does not necessarily represent the exact recording duration but represents the scheduled recording duration. This property does not account for any recording time adjustments such as scheduledDurationAdjust, and device latencies.

Default Value: N/A – Required on input.
Sort Order: Property Specific, based on elapsed time. Ascending: shortest elapsed time first.
Input: The desired setting.
Output: The current setting.

### B.4.5 scheduledProgramCode

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduledProgramCode</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.5.1</td>
</tr>
</tbody>
</table>

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

**Description:** The `scheduledProgramCode` property indicates the program code provided by a program guide service of a particular program item. The format of the program code is defined by the program guide service. A `scheduledProgramCode@type` property MUST be specified with this property to identify the program guide service used.

**Default Value:** N/A – Required on input.

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

**Input:** The desired setting.

**Output:** The current setting.

---

### B.4.5.1 scheduledProgramCode@type

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduledProgramCode@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.5.1.1</td>
</tr>
</tbody>
</table>

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

**Description:** The `scheduledProgramCode@type` property indicates the type of the program guide service that defines the program code specified in the `scheduledProgramCode` property. The format of this property is “<ICANN registered domain>"_"<program code name>".”

**Example:** “epg.com_GuideCode”.

**Default Value:** N/A – Required on input.

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

**Input:** The desired setting.

**Output:** The current setting.

---

### B.5 Matching Content Criteria Properties

**Table B-14: Matching Content Criteria Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>matchingName</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.5.1</td>
</tr>
<tr>
<td>matchingName@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.5.1.1</td>
</tr>
<tr>
<td>matchingName@subStringMatch</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.5.1.2</td>
</tr>
<tr>
<td>matchingID</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.5.2</td>
</tr>
<tr>
<td>matchingID@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.5.2.1</td>
</tr>
</tbody>
</table>

---

### B.5.1 matchingName

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

**Description:** The `matchingName` property contains (part of) the name of a program or series. To match the criteria of this `recordSchedule`, an external item’s name information MUST (partially) match the specified
**matchingName** value. Matching SHOULD be done using lexical matching (see Section 2.2.2.27, “Lexical Matching”). It MAY be done using simple non-case-sensitive matching (see Section 2.2.2.29, “Simple Non-case-sensitive Matching”).

Example: “NFL Worldcup 2005”, “Friends”.

**Default Value**: N/A – Required on input.

**Sort Order**: Lexical.

Note: This is an exception to the normal rule of type Relationship sorting. The equivalent of type Relationship sorting may be achieved by including “+srs:matchingName@type” in the sort property list immediately in front of “+srs:matchingName”.

**Input**: The desired setting.

**Output**: The current setting.

### B.5.1.1 matchingName@type

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

**Description**: IF set to “PROGRAM”, then the **matchingName** property contains a program name. If set to “SERIES”, then the **matchingName** property contains a series name.

**Default Value**: N/A – Required on input.

**Sort Order**: Property Specific, based on the order in Table B-15. Ascending: first table entry first.

**Input**: The desired setting.

**Output**: The current setting.

#### B.5.1.1.1 allowedValueList for the matchingName@type Property

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“PROGRAM”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“SERIES”</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### B.5.1.2 matchingName@subStringMatch

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

**Description**: If set to “1”, the value specified in the **matchingName** property is used for a substring match search within the program or series name (title). If set to “0” the value specified in the **matchingName** property must match the program or series name exactly.

**Default Value**: “1”.

**Sort Order**: Boolean.

**Input**: The desired setting.

**Output**: The current setting.

### B.5.2 matchingID

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

---

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
Description: The *matchingID* property contains the unique ID of a program or series. To match the criteria of this *recordSchedule*, an external item’s ID information MUST match the specified *matchingID* value.

If the *matchingID* property is set to “*SI_PROGRAMID*”, then the *matchingID* property is formatted as follows:

“<Network ID>,<Transport Stream ID>,<Service ID>,<Program ID>”.

If the *matchingID* property is set to “*SI_SERIESID*”, then the *matchingID* property is formatted as follows:

“<Network ID>,<Transport Stream ID>,<Service ID>,<Series ID>”.

If the *matchingID* property is set to <ICANN Name>, then the *matchingID* property is formatted as follows:

“<Unique content ID, defined by the data provider>”.

Default Value: N/A – Required on input.

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

Input: The desired setting.

Output: The current setting.

B.5.2.1 *matchingID* Property

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

Description: The *matchingID* property indicates the type of the ID that is contained in the *matchingID* property.

Default Value: N/A – Required on input.

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

Input: The desired setting.

Output: The current setting.

B.5.2.1.1 *allowedValueList* for the *matchingID* Property

Table B-16: *allowedValueList* for the *matchingID* Property

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“<em>SI_PROGRAMID</em>”</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>“<em>SI_SERIESID</em>”</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>
| <ICANN Name>_<Identifier> | O   | <ICANN Name>: The ICANN name of the organization that defines the format and values of the *matchingID* property.  
<Identifier>: A unique identifier for the particular ID type, defined by that organization.  
Examples: “mycompany.com_ID1”, “upnp.org_SpecialID”. |
B.6 Matching Qualifying Criteria Properties

Table B-17: Matching Qualifying Criteria Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>matchingChannelID</td>
<td>srs</td>
<td>xsd:string</td>
<td>YES</td>
<td>Appendix B.6.1</td>
</tr>
<tr>
<td>matchingChannelID@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.6.1.1</td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkName</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.6.1.2</td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkID</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.6.1.3</td>
</tr>
<tr>
<td>matchingStartDateTimeRange</td>
<td>srs</td>
<td>xsd:string</td>
<td>YES</td>
<td>Appendix B.6.2</td>
</tr>
<tr>
<td>matchingDurationRange</td>
<td>srs</td>
<td>xsd:string</td>
<td>YES</td>
<td>Appendix B.6.3</td>
</tr>
<tr>
<td>matchingRatingLimit</td>
<td>srs</td>
<td>xsd:string</td>
<td>YES</td>
<td>Appendix B.6.4</td>
</tr>
<tr>
<td>matchingRatingLimit@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.6.4.2</td>
</tr>
<tr>
<td>matchingEpisodeType</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.6.5</td>
</tr>
</tbody>
</table>

B.6.1 matchingChannelID

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

Description: The matchingChannelID property contains a scheduledChannelID value. Its format depends on the matchingChannelID@type property. To match the criteria of this recordSchedule, an external item’s channel information (after translation into the format of a scheduledChannelID property) MUST match one of the specified matchingChannelID values. If this property is omitted from the recordSchedule, the external item’s channel information is not taken into consideration to determine a match.

Default Value: N/A – Not used if omitted on input.

Sort Order: Same as scheduledChannelID.

Input: The desired setting.

Output: The current setting if specified on input. Otherwise not present.

B.6.1.1 matchingChannelID@type

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

Description: The matchingChannelID@type property determines the format that is used for the matchingChannelID property as defined in Appendix B.4.2, “scheduledChannelID” and Appendix B.4.2.1, “scheduledChannelID@type”.

Default Value: N/A – Not used if omitted on input.

Sort Order: Same as scheduledChannelID@type.

Input: The desired setting.

Output: The current setting if specified on input. Otherwise not present.

B.6.1.2 matchingChannelID@distriNetworkName

Namespace: srs  
Property Data Type: xsd:string  
Multi-Valued: NO
**Description:** The \textit{matchingChannelID@distriNetworkname} property definition is identical to the \textit{scheduledChannelID@distriNetworkName} property definition. See Appendix B.4.2.2, “\textit{scheduledChannelID@distriNetworkName}” for details.

**Default Value:** N/A – Not used if omitted on input.

**Sort Order:** Same as \textit{scheduledChannelID@distriNetworkName} property.

**Input:** The desired setting.

**Output:** The current setting if specified on input. Otherwise not present.

### B.6.1.3 \textit{matchingChannelID@distriNetworkID}

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

**Description:** The \textit{matchinChannelID@distriNetworkID} property definition is identical to the \textit{scheduledChannelID@distriNetwork} property definition. See Appendix B.4.2.3, “\textit{scheduledChannelID@distriNetworkID}” for details.

**Default Value:** N/A – Not used if omitted on input.

**Sort Order:** Same as \textit{scheduledChannelID@distriNetworkID} property.

**Input:** The desired setting.

**Output:** The current setting if specified on input. Otherwise not present.

### B.6.2 \textit{matchingStartDateTimeRange}

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

**Description:** The \textit{matchingStartDateTimeRange} property contains a date range. The \textit{start-range} format syntax of the \textit{matchingStartDateTimeRange} property is defined in Appendix D, “EBNF Syntax Definitions”.

The value specified after the “/” MUST be equal or greater than the value specified before the “/”.

To match the criteria of this \textit{recordSchedule}, an external item’s start date and time information MUST fall within one of the specified \textit{matchingStartDateTimeRange} ranges. If this property is omitted from the \textit{recordSchedule}, the external item’s start date and time information is not taken into consideration to determine a match.

Note: The \textit{matchingStartDateTimeRange} property is different from the \textit{activePeriod} property in that the first identifies the actual matching criteria whereas the second identifies the period of time when potential matches are to be examined.

**Default Value:** N/A – Not used if omitted on input.

**Sort Order:** Sequenced Sort of two date&time subvalues separated by “/”. Both subvalues are sorted in chronological order.

**Input:** The desired setting.

**Output:** The current setting if specified on input. Otherwise not present.

### B.6.3 \textit{matchingDurationRange}

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

The \textit{matchingDurationRange} property contains a duration range. The \textit{duration-range} format syntax of the \textit{matchingDurationRange} property is defined in Appendix D, “EBNF Syntax Definitions”.

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
The value specified after the “/” MUST be equal or greater than the value specified before the “/”.

To match the criteria of this recordSchedule, an external item’s duration information (after translation into the format of a scheduledDuration property) MUST fall within the specified matchingDurationRange range. If this property is omitted from the recordSchedule, the external item’s duration information is not taken into consideration to determine a match.

Default Value: N/A – Not used if omitted on input.

Sort Order: Sequenced Sort of two duration subvalues separated by “/”.

Input: The desired setting.

Output: The current setting if specified on input. Otherwise not present.

B.6.4 matchingRatingLimit

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

Description: The matchingRatingLimit property indicates a maximum allowed rating. Several different rating systems are available. The rating system is indicated in the matchingRatingLimit@type property. The allowed values for the matchingRatingLimit property depend on the rating system used.

Common rating systems as well as their allowed rating values (in order of ascending restriction level beginning with the most lenient) for each rating system are defined below.

Other values MAY be specified using other rating systems identified by their ICANN domain names.

To match the criteria of this recordSchedule, an external item’s rating information MUST be less than or equal to all of the specified matchingRatingLimit values. If this property is omitted from the recordSchedule, the external item’s rating information is not taken into consideration to determine a match.

If the external item does not contain rating information and this property is specified, the external item will not be recorded.

Default Value: N/A – Not used if omitted on input.

Sort Order: type Relationship.

For each value of matchingRatingLimit@type, based on the order in the table associated with the matchingRatingLimit@type property below. Ascending: first table entry first.

Input: The desired setting.

Output: The current setting if specified on input. Otherwise not present.

B.6.4.1 allowedValueList for the matchingRatingLimit Property

The allowed values for the matchingRatingLimit property depends on the matchingRatingLimit@type property. The following tables list the allowed values for each defined rating system.

Table B-18: allowedValueList for the matchingRatingLimit Property Using the MPAA Rating System (matchingRatingLimit@type = “MPAA.ORG”)

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“G”</td>
<td>R</td>
<td>General Audiences.</td>
</tr>
<tr>
<td>“PG”</td>
<td>R</td>
<td>Parental Guidance Suggested.</td>
</tr>
<tr>
<td>“PG-13”</td>
<td>R</td>
<td>Parents Strongly Cautioned.</td>
</tr>
<tr>
<td>“R”</td>
<td>R</td>
<td>Restricted.</td>
</tr>
<tr>
<td>“NC-17”</td>
<td>R</td>
<td>No One 17 and Under Admitted.</td>
</tr>
</tbody>
</table>
Table B-19: allowedValueList for the matchingRatingLimit Property Using the RIAA Rating System (matchingRatingLimit@type = “RIAA.ORG”)

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“NR”</td>
<td>R</td>
<td>Not Rated Yet.</td>
</tr>
</tbody>
</table>

Table B-20: allowedValueList for the matchingRatingLimit Property Using the ESRB Rating System (matchingRatingLimit@type = “ESRB.ORG”)

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“EC”</td>
<td>R</td>
<td>Early Childhood.</td>
</tr>
<tr>
<td>“E”</td>
<td>R</td>
<td>Everyone.</td>
</tr>
<tr>
<td>“E10+”</td>
<td>R</td>
<td>Everyone 10 and Older.</td>
</tr>
<tr>
<td>“T”</td>
<td>R</td>
<td>Teen.</td>
</tr>
<tr>
<td>“M”</td>
<td>R</td>
<td>Mature.</td>
</tr>
<tr>
<td>“AO”</td>
<td>R</td>
<td>Adults Only.</td>
</tr>
<tr>
<td>“RP”</td>
<td>R</td>
<td>Rating Pending.</td>
</tr>
</tbody>
</table>

Table B-21: allowedValueList for the matchingRatingLimit Property Using the TVGUIDELINES Rating System (matchingRatingLimit@type = “TVGUIDELINES.ORG”)

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“TV-Y”</td>
<td>R</td>
<td>All Children.</td>
</tr>
<tr>
<td>“TV-Y7”</td>
<td>R</td>
<td>Directed to Older Children.</td>
</tr>
<tr>
<td>“TV-Y7FV”</td>
<td>R</td>
<td>Directed to Older Children – Fantasy Violence.</td>
</tr>
<tr>
<td>“TV-G”</td>
<td>R</td>
<td>General Audience.</td>
</tr>
<tr>
<td>“TV-PC”</td>
<td>R</td>
<td>Parental Guidance Suggested.</td>
</tr>
<tr>
<td>“TV-14”</td>
<td>R</td>
<td>Parents Strongly Cautioned.</td>
</tr>
<tr>
<td>“TV-MA”</td>
<td>R</td>
<td>Mature Audience Only.</td>
</tr>
</tbody>
</table>

B.6.4.2 matchingRatingLimit@type

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

Description: The matchingRatingLimit@type property indicates the rating system used in the matchingRatingLimit property. Several different rating systems are available. The allowed values for the matchingRatingLimit property depend on the rating system used.

Other rating systems MAY be specified using their ICANN domain names.

This is not a matching property. It is used in conjunction with the matchingRatingLimit property and identifies the used rating system.
Default Value: N/A – Required in input.

Sort Order: Lexical.

Input: The desired setting.

Output: The current setting.

### B.6.4.2.1 allowedValueList for the `matchingRatingLimit@type` Property

**Table B-22: allowedValueList for the `matchingRatingLimit@type` Property**

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>“MPAA.ORG”</td>
<td>O</td>
<td>The Motion Picture Association of America.</td>
<td>At least one value in these rows MUST be supported by a compliant ScheduledRecording service implementation. Control points should support all values in these rows.</td>
</tr>
<tr>
<td>“RIAA.ORG”</td>
<td>O</td>
<td>The Recording Industry Association of America.</td>
<td></td>
</tr>
<tr>
<td>“ESRB.ORG”</td>
<td>O</td>
<td>The Entertainment Software Rating Board.</td>
<td></td>
</tr>
<tr>
<td>“TVGUIDELINES.ORG”</td>
<td>O</td>
<td>TV Parental Guidelines.</td>
<td></td>
</tr>
<tr>
<td>&lt;ICANN Name&gt;_&lt;Identifier&gt;</td>
<td>X</td>
<td>&lt;ICANN Name&gt;: The ICANN name of the organization that defines the rating. &lt;Identifier&gt;: A unique identifier for a particular rating system, defined by that organization. Examples: “mycompany.com_RS1”, “upnp.org_ratingx”.</td>
<td></td>
</tr>
</tbody>
</table>

### B.6.5 `matchingEpisodeType` Property

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

Description: The `matchingEpisodeType` property indicates the type of content to be recorded in terms of its broadcast novelty. To match the criteria of this `recordSchedule`, an external item’s episode type information MUST match the specified `matchingEpisodeType` value. If this property is omitted from the `recordSchedule`, the external item’s episode type information is not taken into consideration to determine a match. If the external item does not contain episode type information and this property is specified, the external item will not be recorded.

Default Value: N/A – Not used if omitted on input.

Sort Order: Property Specific, based on the order in Table B-23. Ascending: first table entry first.

Input: The desired setting.

Output: The current setting if specified on input. Otherwise not present.
B.6.5.1 allowedValueList for the matchingEpisodeType Property

Table B-23: allowedValueList for the matchingEpisodeType Property

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ALL&quot;</td>
<td>R</td>
<td>All programs are recorded.</td>
</tr>
<tr>
<td>&quot;FIRST-RUN&quot;</td>
<td>R</td>
<td>Only programs that have an original air date equal to the current date are recorded.</td>
</tr>
<tr>
<td>&quot;REPEAT&quot;</td>
<td>R</td>
<td>Only programs that have an original air date earlier than the current date are recorded.</td>
</tr>
</tbody>
</table>

B.7 Content Control Properties

Table B-24: Content Control Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalDesiredRecordTasks</td>
<td>srs</td>
<td>xsd:unsignedInt</td>
<td>NO</td>
<td>Appendix B.7.1</td>
</tr>
<tr>
<td>scheduledStartDateAdjust</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.7.2</td>
</tr>
<tr>
<td>scheduledDurationAdjust</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.7.3</td>
</tr>
<tr>
<td>activePeriod</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.7.4</td>
</tr>
<tr>
<td>durationLimit</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.7.5</td>
</tr>
<tr>
<td>durationLimit@effect</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.7.5.1</td>
</tr>
<tr>
<td>channelMigration</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.7.6</td>
</tr>
<tr>
<td>timeMigration</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.7.7</td>
</tr>
<tr>
<td>allowDuplicates</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.7.8</td>
</tr>
</tbody>
</table>

B.7.1 totalDesiredRecordTasks

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

Description: The totalDesiredRecordTasks property indicates the maximum number of recordTask instances, associated with a given recordSchedule that will ever be generated over the lifetime of the recordSchedule. A value of 0 means that an unlimited number of recordTask instances can be spawned from the recordSchedule.

This property is used to enable or disable recurrence. If a value different from 1 is specified in the totalDesiredRecordTasks property, then the recordSchedule MUST remain active after the first recordTask has been spawned and MUST monitor its internal state to determine if the conditions that caused the first recordTask to be spawned are met again in the future. Whenever this happens, a new recordTask MUST be spawned until the total number of spawned recordTask instances reaches the value, specified in the totalDesiredRecordTasks property. The activePeriod property can be used to terminate this process prematurely.

Default Value: 1 (recurrence is disabled by default).

Sort Order: Numeric.

Input: The desired setting.

Output: The current setting.
B.7.2 **scheduledStartDateTimeAdjust**

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

Description: The **scheduledStartDateTimeAdjust** property indicates a time period to be applied as an adjustment to the scheduled start time. The duration-adj format syntax of the **scheduledStartDateTimeAdjust** property is defined in Appendix D, “EBNF Syntax Definitions”.

Note that the **scheduledStartDateTimeAdjust** property can take on both positive and negative values. Negative values provide pre-roll functionality (notice the + sign in the formula below) whereas positive values allow for starting the recording a certain period of time into the recording. The actual scheduled start time is calculated as:

\[
\text{actualScheduledStartDateTime} = \text{scheduledStartDateTime} + \text{scheduledStartDateTimeAdjust}
\]

Default Value: Vendor-defined.

Sort Order: Property Specific, based on elapsed time. Ascending: from longest negative elapsed time to longest positive elapsed time.

Input: The desired setting.

Output: The current setting.

---

B.7.3 **scheduledDurationAdjust**

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

Description: The **scheduledDurationAdjust** property indicates a period of time to be applied as an adjustment to the scheduled duration time. The duration-adj format syntax of the **scheduledDurationAdjust** property is defined in Appendix D, “EBNF Syntax Definitions”.

Note that the **scheduledDurationAdjust** property can take on both positive and negative values. Positive values provide post-roll functionality whereas negative values allow for ending the recording a certain time period before the end of the recording. The actual scheduled end time and actual scheduled duration are calculated as:

\[
\text{actualScheduledEndDateTime} = \text{scheduledStartDateTime} + \text{scheduledDuration} + \text{scheduledDurationAdjust}
\]

\[
\text{actualScheduledDuration} = \text{actualScheduledEndDateTime} - \text{actualScheduledStartDateTime} = \text{scheduledDuration} + \text{scheduledDurationAdjust} - \text{scheduledStartDateTimeAdjust}
\]

Default Value: Vendor-defined.

Sort Order: Property Specific, based on elapsed time. Ascending: from longest negative elapsed time to longest positive elapsed time.

Input: The desired setting.

Output: The current setting.

---

B.7.4 **activePeriod**

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

Description: The **activePeriod** property indicates the date&time range within which the **recordSchedule** is active; that is: the **recordSchedule** MUST NOT spawn any **recordTask** instances whose **actualStartDateTime** fall outside the period specified in the **activePeriod** property. The start-range format syntax of the **activePeriod** property is defined in Appendix D, “EBNF Syntax Definitions”.

The value specified after the “/” MUST be equal or greater than the value specified before the “/”.

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
A recordSchedule MUST not generate new recordTask instances for programs broadcast after the expiration date.

Note: The activePeriod property is different from the matchingStartDateDateTimeRange property in that the first identifies the period of time when potential matches are to be examined whereas the second identifies the actual matching criteria.

Default Value: “NOW/INFINITY”.

Sort Order: Sequenced Sort of two date&time subvalues separated by “/”.

Both subvalues are sorted in chronological order.

Input: The desired setting.

Output: The current setting.

B.7.5 durationLimit

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

Description: The durationLimit property indicates the maximum allowed duration of the recording. The duration-long format syntax of the durationLimit property is defined in Appendix D, “EBNF Syntax Definitions”.

If the actual duration of the recording exceeds the value specified in the durationLimit property, then the ScheduledRecording service MUST stop recording and either delete the partially recorded content so far or preserve part of the recorded content depending on the current setting of the durationLimit@effect property.

If the durationLimit property is set to “INFINITY”, then no limit is in effect.

Example: the value “P02:30:00” indicates that the recording MUST be stopped after two and a half hours.

Default Value: Vendor-defined.

Sort Order: Property Specific, based on elapsed time. Ascending: shortest elapsed time first.

Input: The desired setting.

Output: The current setting.

B.7.5.1 durationLimit@effect

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

Description: The durationLimit@effect property determines the behavior of the recordSchedule when the duration of the content exceeds the value specified in the durationLimit property.

If set to “SKIP”, then the partially recorded content is deleted once the actualDuration of the recording exceeds the value specified in the durationLimit property.

If set to “LAST”, then only the latest part (in length equal to the value specified in the durationLimit property) of the content is preserved, effectively deleting the first part of the recording.

If set to “FIRST”, then only the initial part (in length equal to the value specified in the durationLimit property) of the content is preserved, effectively deleting the last part of the recording.

Default Value: Vendor-defined.

Sort Order: Property Specific, based on the order in Table B-25. Ascending: first table entry first.

Input: The desired setting.

Output: The current setting.
B.7.5.1.1 enabledValueList for the durationLimit@effect Property

Table B-25: enabledValueList for the durationLimit@effect Property

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKIP</td>
<td>O</td>
<td>At least one value in these rows MUST be supported. Control points should support all values in these rows.</td>
</tr>
<tr>
<td>LAST</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>FIRST</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>

B.7.6 channelMigration

Namespace: srs  
Property Data Type: xsd:boolean  
Multi-Valued: NO

**Description:** A program’s scheduled channel may change between the time the recordSchedule was created and the actual broadcast time. If this property is set to “1”, then the ScheduledRecording service MUST automatically follow the program if it moves to another channel (The reservation will be tracking broadcast channel change). If this value is set to “0”, then the ScheduledRecording service does not follow the program, and the recording will take place on the channel that was specified at the time the recordSchedule created the associated recordTask.

**Default Value:** Vendor-defined.

**Sort Order:** Boolean.

**Input:** The desired setting.

**Output:** The current setting.

B.7.7 timeMigration

Namespace: srs  
Property Data Type: xsd:boolean  
Multi-Valued: NO

**Description:** A program’s scheduled date&time may change between the time the recordSchedule was created and the actual broadcast time. If this property is set to “1”, then the ScheduledRecording service MUST automatically follow the program if it moves to another date&time (The reservation will be tracking broadcast date&time change). If this value is set to “0”, then the ScheduledRecording service does not follow the program, and the recording will take place at the date&time that was specified at the time the recordSchedule created the associated recordTask.

**Default Value:** Vendor-defined.

**Sort Order:** Boolean.

**Input:** The desired setting.

**Output:** The current setting.

B.7.8 allowDuplicates

Namespace: srs  
Property Data Type: xsd:boolean  
Multi-Valued: NO

**Description:** If set to “1”, then programs are recorded, even if a duplicate program has already been recorded as a result of the recordSchedule. If set to “0”, no duplicates are recorded. Detection of duplicate programs is device- and EPG-dependent.

**Default Value:** Vendor-defined.

**Sort Order:** Boolean.

**Input:** The desired setting.
Output: The current setting.

B.8 Storage Related Properties

Table B-26: Storage Related Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>persistedRecordings</td>
<td>srs</td>
<td>xsd:unsignedInt</td>
<td>NO</td>
<td>Appendix B.8.1</td>
</tr>
<tr>
<td>persistedRecordings@latest</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.8.1.1</td>
</tr>
<tr>
<td>persistedRecordings@preAllocation</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.8.1.2</td>
</tr>
<tr>
<td>persistedRecordings@storedLifetime</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.8.1.3</td>
</tr>
</tbody>
</table>

B.8.1 persistedRecordings

Namespace: srs

Property Data Type: xsd:unsignedInt

Multi-Valued: NO

Description: The persistedRecordings property indicates the minimum number of recordings for a given recordSchedule that will be preserved at all times, once available. Even when the ScheduledRecording service needs to make space for other recordings, this minimum number of recordings (that is: the actual content) generated by the recordSchedule will not be deleted. However, if more recordings, associated with the recordSchedule exist, then these excess recordings MAY be deleted by the ScheduledRecording service. Whether the oldest or the newest excess recordings will be deleted depends on the value of the persistedRecordings@latest property.

Default Value: Vendor-defined.

Sort Order: Numeric.

Input: The desired setting.

Output: The current setting.

B.8.1.1 persistedRecordings@latest

Namespace: srs

Property Data Type: xsd:boolean

Multi-Valued: NO

Description: The persistedRecordings@latest property indicates whether newest or oldest recordings are preserved. If set to “1”, then the newest recordings are preserved. The recordings prior to these MAY be deleted when more recent content is recorded.

If set to “0”, then the oldest recordings are preserved. Older content will never be deleted to make room for newer content.

Default Value: Vendor-defined.

Sort Order: Boolean.

Input: The desired setting.

Output: The current setting.

B.8.1.2 persistedRecordings@preAllocation

Namespace: srs

Property Data Type: xsd:boolean

Multi-Valued: NO
Description: The `persistedRecordings@preAllocation` property indicates whether to reserve storage space on beforehand to accommodate for the number of recordings as indicated by the `persistedRecordings` property. If set to "1", adequate storage space is reserved. To reserve storage space, the ScheduledRecording service calculates a best estimate based on parameters such as record quality, start time and duration adjustment etc. However, the ScheduledRecording service can never guarantee that sufficient storage space is reserved to accommodate the total number of recordings, specified in the `persistedRecordings` property. If set to "0", no storage space is reserved.

Default Value: Vendor-defined.

Sort Order: Boolean.

Input: The desired setting.

Output: The current setting.

B.8.1.3 `persistedRecordings@storedLifetime`

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

Description: The `persistedRecordings@storedLifetime` property indicates the minimum time recorded content associated with a `recordSchedule` will be preserved after the recording completes. This will prohibit a recording from being deleted by the auto-delete operation within the specified time period. The duration-any format syntax of the `persistedRecordings@storedLifetime` property is defined in Appendix D, “EBNF Syntax Definitions”.

If the value is set to “INFINITY”, then the content MUST never be automatically deleted.

A value of "ANY" indicates that the content can be deleted at any time by the auto-delete operation. However, it is RECOMMENDED that a ScheduledRecording service implementation only deletes content when space is needed.

Default Value: Vendor-defined.

Sort Order: Property Specific, based on elapsed time. Ascending: shortest elapsed time first. “ANY” is considered the shortest elapsed time possible; “INFINITY” is considered the longest elapsed time possible.

Input: The desired setting.

Output: The current setting.

B.9 Schedule State Properties

Table B-27: Schedule State Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleState</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.9.1</td>
</tr>
<tr>
<td>scheduleState@currentErrors</td>
<td>srs</td>
<td>CSV (xsd:int)</td>
<td>NO</td>
<td>Appendix B.9.1.2</td>
</tr>
<tr>
<td>abnormalTasksExist</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.9.2</td>
</tr>
</tbody>
</table>

B.9.1 `scheduleState`

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

Description: The `scheduleState` property indicates the overall state of the `recordSchedule` itself.

Default Value: N/A – Output only.

Sort Order: Property Specific, based on the order in Table B-28. Ascending: first table entry first.
Input: N/A.
Output: The current setting.

**B.9.1.1 allowedValueList for the `scheduleState` Property**

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“OPERATIONAL”</td>
<td>R</td>
<td><code>recordSchedule</code> is operating and spawning <code>recordTask</code> instances as scheduled.</td>
</tr>
<tr>
<td>“COMPLETED”</td>
<td>R</td>
<td><code>recordSchedule</code> is completed and reached final disposition. No properties will change.</td>
</tr>
<tr>
<td>“ERROR”</td>
<td>R</td>
<td><code>recordSchedule</code> ceases spawning <code>recordTask</code> instances due to error.</td>
</tr>
</tbody>
</table>

**B.9.1.2 `scheduleState@currentErrors`**

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

Description: The `scheduleState@currentErrors` property indicates the current error(s) that cause the schedule to be in the “ERROR” state. This error list pertains specifically to the behavior of a `recordSchedule` and describes a `recordSchedule`'s inability to create new tasks. When the `scheduleState` property has the value “OPERATIONAL”, the `scheduleState@currentErrors` property MUST be empty. The list of error codes are listed in the `recordSchedule` error code section.

Default Value: N/A – Output only.
Sort Order: Sequenced Numeric.
Input: N/A.
Output: The current setting.

**B.9.1.2.1 allowedValueList for the `scheduleState@currentErrors` Property**

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-99</td>
<td>N/A</td>
<td>Reserved</td>
</tr>
<tr>
<td>100</td>
<td>R</td>
<td>General error – an error is detected but the cause can not be identified.</td>
</tr>
<tr>
<td>101</td>
<td>O</td>
<td>The number of spawned <code>recordTask</code> instances has reached some device dependent limit.</td>
</tr>
<tr>
<td>102</td>
<td>O</td>
<td>EPG information not available.</td>
</tr>
<tr>
<td>103</td>
<td>O</td>
<td><code>recordSchedule</code> is disabled by the user.</td>
</tr>
<tr>
<td>104</td>
<td>O</td>
<td>Insufficient memory – The system does not have enough system memory to create any additional <code>recordTask</code> instances.</td>
</tr>
<tr>
<td>105</td>
<td>O</td>
<td>General resource error – some system related resource is causing the <code>recordSchedule</code> to malfunction.</td>
</tr>
<tr>
<td>106-149</td>
<td>O</td>
<td>Reserved for future <code>recordSchedule</code> error codes.</td>
</tr>
<tr>
<td>150-199</td>
<td>X</td>
<td>Vendor extended <code>recordSchedule</code> error codes.</td>
</tr>
<tr>
<td>Value</td>
<td>R/O</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----</td>
<td>------------------------------</td>
</tr>
<tr>
<td>200 and above</td>
<td>N/A</td>
<td>Reserved for future extensions.</td>
</tr>
</tbody>
</table>

**B.9.2 abnormalTasksExist**

Namespace: srs  
Property Data Type: xsd:boolean  
Multi-Valued: NO

**Description:** If this property is set to "1", then that indicates that at least one abnormal recordTask exists for the recordSchedule. If this property is set to "0", then no abnormal recordTask exists for the recordSchedule. A recordTask is considered abnormal if it reaches any state other than "IDLE.READY", "ACTIVE.RECORDING.FROMSTART.OK" or "DONE.FULL".

Default Value: N/A – Output only.

Sort Order: Boolean.

Input: N/A.

Output: The current setting.

**B.10 Statistics Properties**

Table B-30: Statistics Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>currentRecordTaskCount</td>
<td>srs</td>
<td>xsd:unsignedInt</td>
<td>NO</td>
<td>Appendix B.10.1</td>
</tr>
<tr>
<td>totalCreatedRecordTasks</td>
<td>srs</td>
<td>xsd:unsignedInt</td>
<td>NO</td>
<td>Appendix B.10.2</td>
</tr>
<tr>
<td>totalCompletedRecordTasks</td>
<td>srs</td>
<td>xsd:unsignedInt</td>
<td>NO</td>
<td>Appendix B.10.3</td>
</tr>
</tbody>
</table>

**B.10.1 currentRecordTaskCount**

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

**Description:** The currentRecordTaskCount property indicates the number of existing recordTask instances that are currently associated with a given recordSchedule. Previously generated recordTask instances that have finished recording and that have been (auto-)deleted by the ScheduledRecording service are not taken into account.

Default Value: N/A – Output only.

Sort Order: Numeric.

Input: N/A.

Output: The current setting.

**B.10.2 totalCreatedRecordTasks**

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

**Description:** The totalCreatedRecordTasks property indicates how many recordTask instances have been created during the lifetime of the associated recordSchedule. This includes previously generated recordTask instances that have finished recording and that have been (auto-)deleted by the ScheduledRecording service.

Default Value: N/A – Output only.

Sort Order: Numeric.
Input: N/A.
Output: The current setting.

### B.10.3 **totalCompletedRecordTasks**

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

**Description:** The `totalCompletedRecordTasks` property indicates how many `recordTask` instances have been completed (that is: reached any of the “DONE.xxx” states, during the lifetime of the associated `recordSchedule`). This includes previously generated `recordTask` instances that have finished recording and that have been (auto-)deleted by the ScheduledRecording service.

**Default Value:** N/A – Output only.

**Sort Order:** Numeric.

**Input:** N/A.
**Output:** The current setting.

### B.11 Task General Properties

**Table B-31: Task General Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordScheduleID</td>
<td>srs</td>
<td>xsd:string</td>
<td><strong>NO</strong></td>
<td>Appendix B.11.1</td>
</tr>
<tr>
<td>recordedCDSObjectID</td>
<td>srs</td>
<td>xsd:string</td>
<td><strong>NO</strong></td>
<td>Appendix B.11.2</td>
</tr>
<tr>
<td>recordedCDSObjectID@link</td>
<td>srs</td>
<td>xsd:string</td>
<td><strong>NO</strong></td>
<td>Appendix B.11.2.1</td>
</tr>
</tbody>
</table>

#### B.11.1 **recordScheduleID**

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

**Description:** The `recordScheduleID` property contains the value of the `@id` property of the `recordSchedule` that generated the `recordTask`.

**Default Value:** N/A – Output only.

**Sort Order:** Lexical or Lexical Numeric.

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

**Input:** N/A.
**Output:** The current setting.

#### B.11.2 **recordedCDSObjectID**

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

**Description:** The `recordedCDSObjectID` property contains the `didl-lite:@id` property value of the ContentDirectory service object that represents the content recorded by the `recordTask`.

**Default Value:** N/A – Output only.

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

Input: N/A.

Output: The current setting.

B.11.2.1  recordedCDSObjectID@link
Namespace: srs  Property Data Type: xsd:string  Multi-Valued: NO

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

Default Value: N/A – Output only.

Sort Order: Same as cdsReference@link property.

Input: N/A.

Output: The current setting.

B.12  Task Content Identification Properties

Table B-32:  Task Content Identification Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskCDSObjectID</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.12.1</td>
</tr>
<tr>
<td>taskCDSObjectID@link</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.12.1.1</td>
</tr>
<tr>
<td>taskChannelID</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.12.2</td>
</tr>
<tr>
<td>taskChannelID@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.12.2.1</td>
</tr>
<tr>
<td>taskChannelID@distriNetworkName</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.12.2.2</td>
</tr>
<tr>
<td>taskChannelID@distriNetworkID</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.12.2.3</td>
</tr>
<tr>
<td>taskStartDateTime</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.12.2.2</td>
</tr>
<tr>
<td>taskDuration</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.12.4</td>
</tr>
<tr>
<td>taskProgramCode</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.12.5</td>
</tr>
<tr>
<td>taskProgramCode@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.12.5.1</td>
</tr>
<tr>
<td>recordQuality</td>
<td>srs</td>
<td>xsd:string</td>
<td>YES</td>
<td>Appendix B.12.6</td>
</tr>
<tr>
<td>recordQuality@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.12.6.2</td>
</tr>
</tbody>
</table>

B.12.1  taskCDSObjectID
Namespace: srs  Property Data Type: xsd:string  Multi-Valued: NO

Description: The taskCDSObjectID property contains the didl-lite:@id property value of the ContentDirectory service object from which relevant metadata information was extracted to create the recordSchedule that generated this recordTask.

Default Value: N/A – Output only.
Sort Order: Lexical or Lexical Numeric.

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

Input: N/A.

Output: The current setting.

B.12.1.1 taskCDSObjectID@link

Namespace: srs

Property Data Type: xsd:string

Multi-Valued: NO

Description: The taskCDSObjectID@link contains a unique, vendor-defined link identifier that unambiguously links its taskCDSObjectID property to a particular edsReference property instance within the same recordTask object. See Appendix B.17, “ContentDirectory Service Imported Properties” for details.

Default Value: N/A – Output only.

Sort Order: Same as edsReference@link property.

Input: N/A.

Output: The current setting.

B.12.2 taskChannelID

Namespace: srs

Property Data Type: xsd:string

Multi-Valued: NO

Description: The taskChannelID property indicates the actual channel that is used for the recording. Its format depends on the taskChannelID@type property. The possible formats and the dependency on the taskChannelID@type property are identical to the possible formats of the scheduledChannelID and its dependency on the scheduledChannelID@type property as described in Appendix B.4.2, “scheduledChannelID” and Appendix B.4.2.1, “scheduledChannelID@type”.

Default Value: N/A – Output only.

Sort Order: Same as scheduledChannelID property.

Input: N/A.

Output: The current setting.

B.12.2.1 taskChannelID@type

Namespace: srs

Property Data Type: xsd:string

Multi-Valued: NO

Description: The taskChannelID@type property determines the format that is used for the taskChannelID property as defined above. See Appendix B.4.2.1, “scheduledChannelID@type” for details and allowed values.

Default Value: N/A – Output only.

Sort Order: Same as scheduledChannelID@type property.

Input: N/A.

Output: The current setting.

B.12.2.2 taskChannelID@distriNetworkName

Namespace: srs

Property Data Type: xsd:string

Multi-Valued: NO
Description: The taskChannelID@distriNetworkName property definition is identical to the scheduledChannelID@distriNetworkName property definition. See Appendix B.4.2.2, “scheduledChannelID@distriNetworkName” for details.

Default Value: N/A – Output only.

Sort Order: Same as scheduledChannelID@distriNetworkName property.

Input: N/A.

Output: The current setting.

B.12.2.3 taskChannelID@distriNetworkID

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

Description: The taskChannelID@distriNetworkID property definition is identical to the scheduledChannelID@distriNetwork property definition. See Appendix B.4.2.3, “scheduledChannelID@distriNetworkID” for details.

Default Value: N/A – Output only.

Sort Order: Same as scheduledChannelID@distriNetworkID property.

Input: N/A.

Output: The current setting.

B.12.3 taskStartDateTime

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

Description: The taskStartDateTime property indicates the actual start date&time (based on the current information) of the recording. This date&time does not include any adjustments. These are reflected in the taskStartDateTimeAdjust property. The date-time format syntax of the taskStartDateTime property is defined in Appendix D, “EBNF Syntax Definitions”.

Default Value: N/A – Output only.

Sort Order: Property Specific, in chronological order.

Input: N/A.

Output: The current setting.

B.12.4 taskDuration

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

Description: The taskDuration property indicates the actual duration (based on the current information) of the recording. This duration does not include any adjustments. These are reflected in the taskDurationAdjust property. The duration format syntax of the taskDuration property is defined in Appendix D, “EBNF Syntax Definitions”.

Default Value: N/A – Output only.

Sort Order: Property Specific, based on elapsed time. Ascending: shortest elapsed time first.

Input: N/A.

Output: The current setting.
B.12.5  **taskProgramCode**  
Namespace: srs  
Property Data Type: xsd:string  
Multi-Valued: NO  

**Description:** The `taskProgramCode` property indicates the actual program code that is used for the recording. The format is identical to the format of the `scheduledProgramCode` property. See Appendix B.4.5, "`scheduledProgramCode`" for details.  

**Default Value:** N/A – Output only.  

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

**Input:** N/A.  
**Output:** The current setting.  

B.12.5.1  **taskProgramCode@type**  
Namespace: srs  
Property Data Type: xsd:string  
Multi-Valued: NO  

**Description:** The `taskProgramCode@type` property indicates the type of the program guide service that defines the program code specified in the `taskProgramCode` property. The format is identical to the format of the `scheduledProgramCode@type` property. See Appendix B.4.5.1, "`scheduledProgramCode@type`" for details.  

**Default Value:** N/A – Output only.  

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

**Input:** N/A.  
**Output:** The current setting.  

B.12.6  **recordQuality**  
Namespace: srs  
Property Data Type: xsd:string  
Multi-Valued: YES  

**Description:** The `recordQuality` property expresses the recording quality level that is used for a particular `recordTask`.  

When the `recordTask` is in the “**IDLE**” phase (the actual recording has not started yet), this property contains a best-known estimate of the recording quality for the recording. Therefore, the `recordQuality` property MUST contain one of the vendor-defined values supported by the ScheduledRecording service. The value “**AUTO**” is not allowed. If the implementation does not have enough information to generate a value with some accuracy, then the value “**UNKNOWN**” MUST be used.  

When the `recordTask` is in the “**ACTIVE**” or “**DONE**” phase, the `recordQuality` property MUST contain one of the values supported by the implementation, that describes the actual recording quality. The values “**AUTO**” and “**UNKNOWN**” are not allowed.  

There are many ways to express recording quality. Some implementations use bitrates, some use user-friendly labels etc. Some implementations might even support multiple ways to express recording quality simultaneously. The `recordQuality` property is used in conjunction with the `recordQuality@type` to allow implementations to express these type variations.  

For each type variation, the allowed values for the `recordQuality` property MUST be the same as the allowed values supported for the corresponding type variation of the `desiredRecordQuality` property, except that “**UNKNOWN**” replaces “**AUTO**”.  

Note that the `recordQuality` property is a multi-valued property. Therefore, the actual recording quality level can be expressed using different type variations simultaneously. As a baseline, all implementations MUST support type variation “**DEFAULT**”. All record quality levels expressed in a certain type variation MUST have equivalent quality levels expressed in all other type variations, supported by the
implementation. If an implementation supports multiple type variations to express recording quality, then it MUST provide the recording quality level expressed in all supported type variations.

Example: Assume a (hypothetical) implementation that supports the type variations “DEFAULT”, “ATSC” and “QLEVEL” for the recordQuality@type property. The following table expresses the supported recordQuality property values for those variations and also indicates how the different type variations interrelate for this particular implementation:

Table B-33: recordQuality Example

<table>
<thead>
<tr>
<th>“DEFAULT”</th>
<th>“ATSC”</th>
<th>“QLEVEL”</th>
</tr>
</thead>
<tbody>
<tr>
<td>“HD”</td>
<td>“1080p30”</td>
<td>“Q1”</td>
</tr>
<tr>
<td></td>
<td>“1080p24”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“1080i60”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“720p60”</td>
<td>“Q2”</td>
</tr>
<tr>
<td></td>
<td>“720p30”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“720p24”</td>
<td></td>
</tr>
<tr>
<td>“ED”</td>
<td>“480p60”</td>
<td></td>
</tr>
<tr>
<td>“SD”</td>
<td>“480p30”</td>
<td>“Q3”</td>
</tr>
<tr>
<td></td>
<td>“480p24”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“480i60”</td>
<td></td>
</tr>
<tr>
<td>“UNKNOWN”</td>
<td>“UNKNOWN”</td>
<td>“UNKNOWN”</td>
</tr>
</tbody>
</table>

- Assuming the actual recording quality of a recordTask is “720p60” (as an example), then the recordTask object MUST include three instances of the recordQuality property as illustrated by the following XML fragment:

  `<recordQuality type="DEFAULT">HD</recordQuality>`
  `<recordQuality type="ATSC">720p60</recordQuality>`
  `<recordQuality type="QLEVEL">Q2</recordQuality>`

- Assuming the actual recording quality of a recordTask is “480p60”, then the recordTask object MUST include three instances of the recordQuality property as illustrated by the following XML fragment:

  `<recordQuality type="DEFAULT">ED</recordQuality>`
  `<recordQuality type="ATSC">480p60</recordQuality>`
  `<recordQuality type="QLEVEL">Q3</recordQuality>`

When the ScheduledRecording service responds to a GetAllowedValues() action with recordQuality information, then the allowed values MUST be listed in order of quality from highest quality to lowest.

Default Value: N/A – Output only.

Sort Order: type Relationship.

Input: N/A.

Output: The current setting.
B.12.6.1 allowedValueList for the recordQuality Property

Table B-34: allowedValueList for the recordQuality Property

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“UNKNOWN”</td>
<td>R</td>
<td>The recording quality is unknown by the ScheduledRecording service. Only applicable when the recordTask is in the “IDLE” phase.</td>
</tr>
<tr>
<td>Vendor-defined</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

B.12.6.2 recordQuality@type

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

Description: There are many ways to express recording quality. Some implementations use bitrates, some use user-friendly labels etc. Some implementations might even support multiple ways to express recording quality simultaneously. The recordQuality@type property is used to express which type variation is used in its associated independent recordQuality property. The “DEFAULT” value MUST be supported.

Default Value: N/A – Output only.

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

Input: N/A.

Output: The current setting.

B.13 Task Matched Content Criteria Properties

Table B-35: Task Matched Content Criteria Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>matchedName</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.13.1</td>
</tr>
<tr>
<td>matchedName@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.13.1.1</td>
</tr>
<tr>
<td>matchedID</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.13.2</td>
</tr>
<tr>
<td>matchedID@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.13.2.1</td>
</tr>
</tbody>
</table>

B.13.1 matchedName

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

Description: The matchedName property contains the full matched name of a program or series. This is the full program or series name of the external item that (partially) matched the name specified in the matchingName property of the recordSchedule.

Default Value: N/A – Output only.

Sort Order: Same as matchingName property.

Input: N/A.

Output: The current setting.

B.13.1.1 matchedName@type

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

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
Description: IF set to “PROGRAM”, then the matchedName property contains a program name. If set to “SERIES”, then the matchedName property contains a series name. The format is identical to the format of the matchingName@type property. See Appendix B.5.1.1, “matchingName@type” for details.

Default Value: N/A – Output only.

Sort Order: Same as matchingName@type property.

Input: N/A.

Output: The current setting.

B.13.2 matchedID

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

Description: The matchedID property contains the matched ID of a program or series. This is the ID of the external item that matched the ID specified in the matchingID property of the recordSchedule. The format is identical to the format of the matchingID property. See Appendix B.5.2, “matchingID” for details.

Default Value: N/A – Output only.

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

Input: N/A.

Output: The current setting.

B.13.2.1 matchedID@type

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

Description: The matchedID@type property indicates the type of the ID that is contained in the matchedID property. The format of this property is identical to the format of the matchingID@type property. See Appendix B.5.2.1, “matchingID@type” for details.

Default Value: N/A – Output only.

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

Input: N/A.

Output: The current setting.

B.14 Task Matched Qualifying Criteria Properties

Table B-36: Task Matched Qualifying Criteria Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>matchedRating</td>
<td>srs</td>
<td>xsd:string</td>
<td>[YES]</td>
<td>Appendix B.14.1</td>
</tr>
<tr>
<td>matchedRating@type</td>
<td>srs</td>
<td>xsd:string</td>
<td>[NO]</td>
<td>Appendix B.14.2</td>
</tr>
<tr>
<td>matchedEpisodeType</td>
<td>srs</td>
<td>xsd:string</td>
<td>[NO]</td>
<td>Appendix B.14.3</td>
</tr>
</tbody>
</table>

B.14.1 matchedRating

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

Description: The matchedRating property contains the actual rating of the recording. This is the rating of the external item that matched (was less or equal to) a rating limit specified in one of the

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
matchingRatingLimit properties of the recordSchedule. The format is identical to the format of the matchingRatingLimit property. See Appendix B.6.4, “matchingRatingLimit” for details.

**Default Value:** N/A – Output only.

**Sort Order:** Same as matchingRatingLimit property.

**Input:** N/A.

**Output:** The current setting.

### B.14.2 matchedRating@type

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

**Description:** The matchedRating@type property indicates the rating system used in the matchedRating property. The format is identical to the format of the matchingRatingLimit@type property. See Appendix B.6.4.2, “matchingRatingLimit@type” for details.

**Default Value:** N/A – Output only.

**Sort Order:** Same as matchingRatingLimit@type property.

**Input:** N/A.

**Output:** The current setting.

### B.14.3 matchedEpisodeType

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

**Description:** The matchedEpisodeType property contains the actual episode type of the recording. This is the episode type of the external item that matched episode type specified in the matchingEpisodeType property of the recordSchedule. The format is identical to the format of the matchingEpisodeType property. See Appendix B.6.5, “matchingEpisodeType” for details.

**Default Value:** N/A – Output only.

**Sort Order:** Same as matchingEpisodeType property.

**Input:** N/A.

**Output:** The current setting.

### B.15 Task Matched Content Control Properties

#### Table B-37: Task Matched Content Control Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskStartDateTimeAdjust</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.15.1</td>
</tr>
<tr>
<td>taskDurationAdjust</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.15.2</td>
</tr>
<tr>
<td>taskDurationLimit</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.15.3</td>
</tr>
<tr>
<td>taskDurationLimit@effect</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.15.4</td>
</tr>
<tr>
<td>taskChannelMigration</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.15.5</td>
</tr>
<tr>
<td>taskTimeMigration</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.15.6</td>
</tr>
</tbody>
</table>
B.15.1 **taskStartDateTimeAdjust**

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

Description: The taskStartDateTimeAdjust property is set to the value of the scheduledStartDateTimeAdjust property of the parent recordSchedule. The format is identical to the format of the scheduledStartDateTimeAdjust property. See Appendix B.7.2, “scheduledStartDateTimeAdjust” for details.

Default Value: N/A – Output only.

Sort Order: Same as scheduledStartDateTimeAdjust property.

Input: N/A.

Output: The current setting.

B.15.2 **taskDurationAdjust**

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

Description: The taskDurationAdjust property is set to the value of the scheduledDurationAdjust property of the parent recordSchedule. The format is identical to the format of the scheduledDurationAdjust property. See Appendix B.7.3, “scheduledDurationAdjust” for details.

Default Value: N/A – Output only.

Sort Order: Same as scheduledDurationAdjust property.

Input: N/A.

Output: The current setting.

B.15.3 **taskDurationLimit**

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

Description: The taskDurationLimit property is set to the value of the durationLimit property of the parent recordSchedule. The format is identical to the format of the durationLimit property. See Appendix B.7.5, “durationLimit” for details.

Default Value: N/A – Output only.

Sort Order: Same as durationLimit property.

Input: N/A.

Output: The current setting.

B.15.4 **taskDurationLimit@effect**

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

Description: The taskDurationLimit@effect property is set to the value of the durationLimit@effect property of the parent recordSchedule. The format is identical to the format of the durationLimit@effect property. See Appendix B.7.5.1, “durationLimit@effect” for details.

Default Value: N/A – Output only.

Sort Order: Same as durationLimit@effect property.

Input: N/A.

Output: The current setting.
B.15.5 taskChannelMigration
Namespace: srs  Property Data Type: xsd:boolean  Multi-Valued: NO
Description: The taskChannelMigration property is set to the value of the channelMigration property of the parent recordSchedule. The format is identical to the format of the channelMigration property. See Appendix B.7.6, “channelMigration” for details.
Default Value: N/A – Output only.
Sort Order: Same as channelMigration property.
Input: N/A.
Output: The current setting.

B.15.6 taskTimeMigration
Namespace: srs  Property Data Type: xsd:boolean  Multi-Valued: NO
Description: The taskTimeMigration property is set to the value of the timeMigration property of the parent recordSchedule. The format is identical to the format of the timeMigration property. See Appendix B.7.7, “timeMigration” for details.
Default Value: N/A – Output only.
Sort Order: Same as timeMigration property.
Input: N/A.
Output: The current setting.

B.16 Task State Properties

Table B-38: State Related Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>NS</th>
<th>Data Type</th>
<th>M-Val</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskState</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.16.1</td>
</tr>
<tr>
<td>taskState@phase</td>
<td>srs</td>
<td>xsd:string</td>
<td>NO</td>
<td>Appendix B.16.1.2</td>
</tr>
<tr>
<td>taskState@startDateTimeMet</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.16.1.3</td>
</tr>
<tr>
<td>taskState@endDateTimeMet</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.16.1.4</td>
</tr>
<tr>
<td>taskState@recording</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.16.1.5</td>
</tr>
<tr>
<td>taskState@someBitsRecorded</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.16.1.6</td>
</tr>
<tr>
<td>taskState@someBitsMissing</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.16.1.7</td>
</tr>
<tr>
<td>taskState@firstBitsRecorded</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.16.1.8</td>
</tr>
<tr>
<td>taskState@lastBitsRecorded</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.16.1.9</td>
</tr>
<tr>
<td>taskState@fatalError</td>
<td>srs</td>
<td>xsd:boolean</td>
<td>NO</td>
<td>Appendix B.16.1.10</td>
</tr>
<tr>
<td>taskState@currentErrors</td>
<td>srs</td>
<td>CSV (xsd:int)</td>
<td>NO</td>
<td>Appendix B.16.1.11</td>
</tr>
<tr>
<td>taskState@errorHistory</td>
<td>srs</td>
<td>CSV (xsd:int)</td>
<td>NO</td>
<td>Appendix B.16.1.12</td>
</tr>
<tr>
<td>taskState@pendingErrors</td>
<td>srs</td>
<td>CSV (xsd:int)</td>
<td>NO</td>
<td>Appendix B.16.1.13</td>
</tr>
<tr>
<td>taskState@infoList</td>
<td>srs</td>
<td>CSV (xsd:int)</td>
<td>NO</td>
<td>Appendix B.16.1.14</td>
</tr>
</tbody>
</table>
B.16.1 taskState
Namespace: srs  Property Data Type: xsd:string  Multi-Valued: NO
Description: The taskState property indicates the overall state of the recordTask.
Default Value: N/A – Output only.
Sort Order: Property Specific, based on the order in Table B-39. Ascending: first table entry first.
Input: N/A.
Output: The current setting.

B.16.1.1 allowedValueList for the taskState Property
This section defines the normative allowed values for the taskState property. Each of these values represents a semantically meaningful combination of values for some of the “low-level” state properties (that is: taskState@xxx). Although it is possible to derive the value of the taskState property from some of the “low-level” taskState@xxx properties, the taskState property provides a more convenient mechanism to determine the current state of the recordTask.

The definition of each state is shown in the table below. This table represents the normative definitions of the various states. Although some of the low-level state properties have been declared as optional, their underlying semantics and their significance towards the definition of each valid taskState value is not diminished. Regardless of whether or not a given device is able to expose all of the low-level state properties, every device has a conceptual notion of property semantics. For example, some devices may not be able to support the taskState@lastBitsRecorded property, however, even these devices have an internal concept that the last bits of the content have or have not been recorded.

In some cases, a specific low-level state property does not contribute to the definition of a given state. In other words, the low-level property can have any value without affecting the semantics of the state. This situation is indicated by a “-” in the table entry.

The “Ø” symbol is used to indicate an empty attribute. The “{}” symbol is used when the attribute is not empty.

Following this table, a more intuitive informational description of each state value and their support level is described.

Table B-39: allowedValueList for the taskState Property

<table>
<thead>
<tr>
<th>Value</th>
<th>@phase</th>
<th>@recording</th>
<th>@someBitsRecorded</th>
<th>@someBitsMissing</th>
<th>@fatalError</th>
<th>@currentErrors</th>
<th>@pendingErrors</th>
<th>@errorHistory</th>
<th>@startDateTimeMet</th>
<th>@endDateTimeMet</th>
<th>@firstBitsRecorded</th>
<th>@lastBitsRecorded</th>
</tr>
</thead>
<tbody>
<tr>
<td>“IDLE.READY”</td>
<td>“IDLE”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Ø</td>
<td>Ø</td>
</tr>
<tr>
<td>“IDLE.ATRISK”</td>
<td>“IDLE”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>{}</td>
<td>0</td>
</tr>
<tr>
<td>“ACTIVE.TRANSITION.FROMSTART”</td>
<td>“ACTIVE”</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>“ACTIVE.TRANSITION.RESTART”</td>
<td>“ACTIVE”</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>{}</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>“ACTIVE.RECORDING.FROMSTART.OK”</td>
<td>“ACTIVE”</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>“ACTIVE.RECORDING.FROMSTART.ATRISK”</td>
<td>“ACTIVE”</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>{}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>“ACTIVE.RECORDING.RESTART.OK”</td>
<td>“ACTIVE”</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>{}</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
* Some implementations may not expose these individual properties to the control point. However, in this case, all visible external behavior of the device MUST be as if it implemented all of the properties as specified in the table above.

In the following table, a more intuitive informational description of each state value and its support level is described.

**Table B-40: allowedValueList for the `taskState` Property**

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“IDLE.READY”</td>
<td>R</td>
<td>The <code>recordTask</code> is waiting for the start time to be reached. No errors have been detected.</td>
</tr>
<tr>
<td>“IDLE.ATRISK”</td>
<td>O</td>
<td>The <code>recordTask</code> is waiting for the start time to be reached while some pending errors exist.</td>
</tr>
<tr>
<td>“ACTIVE.TRANSITION.FROMSTART”</td>
<td>O</td>
<td>The device’s record mechanism has been initiated to record the content from its beginning but no actual recording has occurred.</td>
</tr>
<tr>
<td>“ACTIVE.TRANSITION.RESTART”</td>
<td>O</td>
<td>The device’s record mechanism has been re-initiated following some content loss from previous error conditions.</td>
</tr>
<tr>
<td>“ACTIVE.RECORDING.FROMSTART.OK”</td>
<td>R</td>
<td>The device’s record mechanism is currently continuously recording from the beginning. No current or pending errors exist.</td>
</tr>
<tr>
<td>“ACTIVE.RECORDING.FROMSTART.ATRISK”</td>
<td>O</td>
<td>The device’s record mechanism is currently continuously recording from the beginning. Some pending errors are detected.</td>
</tr>
</tbody>
</table>
### Value List

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ACTIVE.RECORDING.RESTART.OK”</td>
<td>O</td>
<td>The device’s record mechanism is currently recording content, following some content loss from previous error conditions. No current or pending errors exist.</td>
</tr>
<tr>
<td>“ACTIVE.RECORDING.RESTART.ATRISK”</td>
<td>O</td>
<td>The device’s record mechanism is currently recording content following some content loss from previous error conditions. One or more pending errors are detected, which will block the recording in the future.</td>
</tr>
<tr>
<td>“ACTIVE.NOTRECORDING”</td>
<td>O</td>
<td>The device’s record mechanism is currently NOT recording content due to one or more error conditions.</td>
</tr>
<tr>
<td>“DONE.FULL”</td>
<td>R</td>
<td>The <code>recordTask</code> has reached its final disposition and no other property or attribute changes will occur. All of the content has been recorded.</td>
</tr>
<tr>
<td>“DONE.PARTIAL”</td>
<td>R</td>
<td>The <code>recordTask</code> has reached its final disposition and no other property or attribute changes will occur. The content is only partially recorded due to error(s).</td>
</tr>
<tr>
<td>“DONE.EMPTY”</td>
<td>R</td>
<td>The <code>recordTask</code> has reached its final disposition and no other property or attribute changes will occur. No content has been recorded at all due to error conditions.</td>
</tr>
</tbody>
</table>

---

### B.16.1.2 `taskState@phase`

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

**Description:** The `taskState@phase` property indicates the current phase of a `recordTask` within its normal lifetime. The following allowed values for this property are sequentially assigned at the appropriate points in time within the `recordTask`’s normal lifetime: “IDLE” → “ACTIVE” → “DONE”. In certain cases, some of the phase values may be skipped, for example, when a fatal error is detected.

**Default Value:** N/A – Output only.

**Sort Order:** Property Specific, based on the order in Table B-41. Ascending: first table entry first.

**Input:** N/A.

**Output:** The current setting.

#### B.16.1.2.1 allowedValueList for the `taskState@phase` Property

**Table B-41:** allowedValueList for the `taskState@phase` Property

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“IDLE”</td>
<td>R</td>
<td>Indicates that the <code>recordTask</code>’s start time has not yet been reached.</td>
</tr>
</tbody>
</table>
### B.16.1.3 taskState@startDateTimeMet

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

**Description:** The `taskState@startDateTimeMet` property indicates whether the `recordTask`'s `actualStartDateTime` has been reached. See Section 2.2, “Terms” for the definition of `actualStartDateTime`.

If a `recordTask` has reached the “DONE” phase, this property indicates the last status before the `recordTask` reaches the “DONE” phase. Note: if the `recordTask` terminates prematurely (that is: reaches the “DONE” phase before the start time is reached, for example, due to a fatal error), this property is not updated.

**Default Value:** N/A – Output only.

**Sort Order:** Boolean.

**Input:** N/A.

**Output:** The current setting.

### B.16.1.4 taskState@endDateTimeMet

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

**Description:** The `taskState@endDateTimeMet` property indicates whether the `recordTask`'s `actualEndDateTime` has been reached. See Section 2.2, “Terms” for the definition of `actualEndDateTime`.

If a `recordTask` has reached the “DONE” phase, this property indicates the last status before the `recordTask` reaches the “DONE” phase. Note: if the `recordTask` terminates prematurely (that is: reaches the “DONE” phase before the end time is reached, for example, due to a fatal error), this property is not updated.

**Default Value:** N/A – Output only.

**Sort Order:** Boolean.

**Input:** N/A.

**Output:** The current setting.

### B.16.1.5 taskState@recording

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

**Description:** The `taskState@recording` property indicates whether one of the device’s record destinations is currently recording the content identified by the `recordTask`.

**Default Value:** N/A – Output only.

---

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
Sort Order: Boolean.
Input: N/A.
Output: The current setting.

B.16.1.6  \textit{taskState@someBitsRecorded}
Namespace: srs  
Property Data Type: xsd:boolean  
Multi-Valued: \textit{NO}
Description: The \textit{taskState@someBitsRecorded} property indicates whether some portion of the content identified by the \textit{recordTask} has been recorded.
Default Value: N/A – Output only.
Sort Order: Boolean.
Input: N/A.
Output: The current setting.

B.16.1.7  \textit{taskState@someBitsMissing}
Namespace: srs  
Property Data Type: xsd:boolean  
Multi-Valued: \textit{NO}
Description: The \textit{taskState@someBitsMissing} property indicates whether some portion of the content identified by the \textit{recordTask} has not been recorded. This property will be “\textit{0}” as long as all the bits that have been available so far have also been recorded.
Default Value: N/A – Output only.
Sort Order: Boolean.
Input: N/A.
Output: The current setting.

B.16.1.8  \textit{taskState@firstBitsRecorded}
Namespace: srs  
Property Data Type: xsd:boolean  
Multi-Valued: \textit{NO}
Description: The \textit{taskState@firstBitsRecorded} property indicates whether the first portion of the content identified by the \textit{recordTask} has been recorded.
Default Value: N/A – Output only.
Sort Order: Boolean.
Input: N/A.
Output: The current setting.

B.16.1.9  \textit{taskState@lastBitsRecorded}
Namespace: srs  
Property Data Type: xsd:boolean  
Multi-Valued: \textit{NO}
Description: The \textit{taskState@lastBitsRecorded} property indicates whether the ending portion of the content identified by the \textit{recordTask} has been recorded.
Default Value: N/A – Output only.
Sort Order: Boolean.
Input: N/A.
Output: The current setting.
B.16.1.10  taskState@fatalError

Namespace: srs  Property Data Type: xsd:boolean  Multi-Valued: NO

Description: The taskState@fatalError property indicates whether a fatal error has occurred. A fatal error is defined to be an error condition that causes the recordTask to terminate before its actualEndDateTime has been reached.

Default Value: N/A – Output only.

Sort Order: Boolean.

Input: N/A.

Output: The current setting.

B.16.1.11  taskState@currentErrors

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

Description: The taskState@currentErrors property identifies the CSV list of errors that are currently preventing the recordTask from recording. An empty CSV list indicates that there are no errors currently blocking the recording. Multiple errors are listed in order of occurrence starting with the oldest error and ending with the most recent.

When the errors are not resolved before reaching the “DONE” phase, they MAY be persisted in the “DONE” phase. If a device persists current errors, the value of this property MUST be set to the value that this property had immediately prior to entering the “DONE” phase. If a device does not persist current errors, the taskState@currentErrors MUST be empty in the “DONE” phase.

By definition, this property MUST be empty while in the “IDLE” phase. The current errors are also copied to the taskState@errorHistory property.

Default Value: N/A – Output only.

Sort Order: Sequenced Numeric.

Input: N/A.

Output: The current setting.

B.16.1.11.1 allowedValueList for the taskState@currentErrors Property and Other Error Properties

The following table defines error codes for all error properties of a recordTask, such as taskState@currentErrors, taskState@errorHistory, etc. to expose error conditions. This error list can be extended in the future or by vendors. The errors are grouped into separate categories and labeled 1xx, 2xx, 3xx, and 4xx groups, each group representing the nature of errors; that is: general errors, media errors, system errors and content errors, respectively. The grouping of error codes allows a control point to be able to understand the nature of errors when an unknown error code (that is: extended specification or vendor extended) is encountered. For example, if an unknown error is labeled 3xx, it can be interpreted by the control point as 300.

Table B-42: allowedValueList for the taskState@xxx Properties

<table>
<thead>
<tr>
<th>Value</th>
<th>R/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-positive</td>
<td>N/A</td>
<td>These error codes are reserved for future use. Control points should gracefully ignore any non-positive error codes.</td>
</tr>
<tr>
<td>001-099</td>
<td>N/A</td>
<td>Reserved.</td>
</tr>
<tr>
<td>Value</td>
<td>R/O</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>100-199</td>
<td>N/A</td>
<td>General Error Code Group - arbitrary errors, which do not belong to other groups.</td>
</tr>
<tr>
<td>100</td>
<td>R</td>
<td>General Problem – a problem is confirmed, but no specific reason can be identified.</td>
</tr>
<tr>
<td>101</td>
<td>O</td>
<td>Disabled - the recordTask is disabled by the user.</td>
</tr>
<tr>
<td>102</td>
<td>O</td>
<td>The recordTask’s enable/disable behavior is overriding the default behavior specified by the associated recordSchedule.</td>
</tr>
<tr>
<td>103-149</td>
<td>N/A</td>
<td>Reserved for future General Error Codes.</td>
</tr>
<tr>
<td>150-199</td>
<td>N/A</td>
<td>Reserved for vendor-defined General Error Codes.</td>
</tr>
<tr>
<td>200-299</td>
<td>N/A</td>
<td>Media Error Code Group - arbitrary media related errors.</td>
</tr>
<tr>
<td>200</td>
<td>O</td>
<td>General Media Problem – some trouble related to media is detected. Replacing the media may likely resolve it.</td>
</tr>
<tr>
<td>201</td>
<td>O</td>
<td>No Media – necessary media is missing from the recording device.</td>
</tr>
<tr>
<td>202</td>
<td>O</td>
<td>Media Write Protect - write access to the recording media is prohibited.</td>
</tr>
<tr>
<td>203</td>
<td>O</td>
<td>Insufficient Media Space - recording media does not have enough available space to complete the recordTask.</td>
</tr>
<tr>
<td>204</td>
<td>O</td>
<td>Media Low Space - the recording media has low available space and the recordTask may fail. The criteria to determine “low space” is vendor dependent and may be independent from the size of the scheduled content to record.</td>
</tr>
<tr>
<td>205-249</td>
<td>N/A</td>
<td>Reserved for future Media ErrorCodes.</td>
</tr>
<tr>
<td>250-299</td>
<td>N/A</td>
<td>Reserved for vendor-defined Media Error Codes.</td>
</tr>
<tr>
<td>300-399</td>
<td>N/A</td>
<td>System Error Code Group - arbitrary system related error.</td>
</tr>
<tr>
<td>300</td>
<td>O</td>
<td>General System Problem – a problem related to the system is detected. It may affect all recordTask instances in the ScheduledRecording service.</td>
</tr>
<tr>
<td>301</td>
<td>O</td>
<td>Insufficient Memory- the system does not have enough system memory to complete the recordTask.</td>
</tr>
<tr>
<td>302</td>
<td>O</td>
<td>Insufficient Processing - the system does not have enough CPU power to execute the recordTask.</td>
</tr>
<tr>
<td>303</td>
<td>O</td>
<td>Low Memory - the system has low available memory and the recordTask may fail. The criteria to determine “low memory” is vendor dependent and may be independent from the size of the scheduled content to record.</td>
</tr>
<tr>
<td>304</td>
<td>O</td>
<td>Low Processing - the system has low available CPU power and the recordTask may fail. The criteria to determine “low processing” is vendor dependent and may be independent from the size of the scheduled content to record.</td>
</tr>
<tr>
<td>305</td>
<td>O</td>
<td>Signal Lost - the system has lost the input signal.</td>
</tr>
<tr>
<td>306</td>
<td>O</td>
<td>Low Signal - The system has low input signal and the recordTask may fail. The criteria to determine “low processing” is vendor dependent.</td>
</tr>
<tr>
<td>307</td>
<td>O</td>
<td>No EPG - the system lost access to the EPG.</td>
</tr>
<tr>
<td>308-349</td>
<td>N/A</td>
<td>Reserved for future System Error Codes.</td>
</tr>
<tr>
<td>Value</td>
<td>R/O</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-------------</td>
</tr>
<tr>
<td>350-399</td>
<td>N/A</td>
<td>Reserved for vendor-defined System Error Codes.</td>
</tr>
<tr>
<td>400-499</td>
<td>N/A</td>
<td>Content Error Code Group - arbitrary errors related to the content program to be recorded.</td>
</tr>
<tr>
<td>400</td>
<td>O</td>
<td>General Content Problem – a problem related to the content is detected. It may be associated with the content that is being recorded.</td>
</tr>
<tr>
<td>401</td>
<td>O</td>
<td>Conflicting Program Loser – there are other conflicting programs with overlapping time period, and the current recordTask is superseded by the conflicting program.</td>
</tr>
<tr>
<td>402</td>
<td>O</td>
<td>Conflicting Program Winner - there are other conflicting programs with overlapping time period, and the current recordTask superseded the conflicting program.</td>
</tr>
<tr>
<td>403</td>
<td>O</td>
<td>PPV (Pay per View) - the content is PPV and some procedures are needed for the recordTask to begin.</td>
</tr>
<tr>
<td>404</td>
<td>O</td>
<td>Content Rescheduled - the originally scheduled content has been preempted.</td>
</tr>
<tr>
<td>405-449</td>
<td>N/A</td>
<td>Reserved for future Content Error Codes.</td>
</tr>
<tr>
<td>450-499</td>
<td>N/A</td>
<td>Reserved for vendor-defined Content Error Codes.</td>
</tr>
<tr>
<td>500 and above</td>
<td>N/A</td>
<td>Reserved for future new category information extensions.</td>
</tr>
</tbody>
</table>

B.16.1.12  taskState@errorHistory

Description: The taskState@errorHistory property identifies the CSV list of errors that have (at any time) prevented the recordTask from completing successfully. This includes both past and current recording errors. Multiple errors are listed in order of occurrence starting with the oldest error and ending with the most recent. An empty list indicates that none of the recordTask’s content has yet been prevented from being recorded. By definition, this list will always be empty while in the “IDLE” phase. Note: Any errors listed in taskState@currentErrors MUST also be copied to and persisted in this property.

Default Value: N/A – Output only.

Sort Order: N/A.

Input: N/A.

Output: The current setting.

B.16.1.12.1 allowedValueList for the taskState@errorHistory Property

See Appendix B.16.1.11.1, “allowedValueList for the taskState@currentErrors Property” for details.

B.16.1.13  taskState@pendingErrors

Description: The taskState@pendingErrors property identifies the CSV list of errors that may prevent the recordTask from completing successfully at some time in the future unless resolved. An empty CSV list means that no pending errors have been detected. The list of errors that the device is able to detect before they actually occur may be obtained via the GetAllowedValues() action.
Those devices that are not able to detect any pending errors before they actually occur MAY always return an empty list for the value of this property. In this case, the value returned by `GetAllowedValues()` for this property MUST also be an empty list.

If any of these pending errors actually occur, they MUST be added to the `taskState@currentErrors` list and `taskState@ErrorHistory` and removed from this list. When the pending errors did not occur, these errors MAY be persisted to the “DONE” phase. If a device does not persist any pending errors that have not occurred yet, then the `taskState@pendingErrors` MUST be empty in the “DONE” phase. Otherwise the value of this property MUST be set to the value that this property had immediately prior to entering the “DONE” phase.

**Default Value:** N/A – Output only.

**Sort Order:** Sequenced Numeric.

**Input:** N/A.

**Output:** The current setting.

### B.16.1.13.1 allowedValueList for the `taskState@pendingErrors` Property

See Appendix B.16.1.11.1, “allowedValueList for the `taskState@currentErrors` Property” for details.

### B.16.1.14 `taskState@infoList`

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

**Description:** The `taskState@infoList` property identifies the CSV list of additional conditions that have been detected but will not block the current `recordTask`, for example, conflict winner.

The list of possible information that the device is able to detect may be obtained via the `GetAllowedValues()` action.

Devices that are not able to detect any additional information MUST always return an empty list. In this case, the value returned by `GetAllowedValues()` for this property MUST also be an empty list.

**Note:** a device can also use `additionalStatusInfo` to expose information in text format.

**Default Value:** N/A – Output only.

**Sort Order:** Sequenced Numeric.

**Input:** N/A.

**Output:** The current setting.

### B.16.1.14.1 allowedValueList for the `taskState@infoList` Property

See Appendix B.16.1.11.1, “allowedValueList for the `taskState@currentErrors` Property” for details.

### B.17 ContentDirectory Service Imported Properties

ContentDirectory service properties are imported through the `cdsReference` multi-valued property. The main reason to import properties (metadata) from ContentDirectory service objects into a `recordSchedule` or `recordTask` object is to make that object self-contained; that is: a control point can retrieve relevant metadata from the ScheduledRecording service object without having to first extract the object IDs of external ContentDirectory service objects and then retrieve the metadata from these objects via additional actions. In addition, even when the referenced object in the ContentDirectory service is deleted, its metadata is still preserved within the ScheduledRecording service. It is the responsibility of the device to maintain consistency between the actual ContentDirectory service object’s metadata and the metadata contained in the corresponding `cdsReference` property.
The *cdsReference* property MUST contain a *valid* (it MUST contain all the REQUIRED properties as dictated by the DIDL-Lite Schema; also, if dependent properties are imported, their independent properties MUST be imported as well.) and properly escaped *DIDL-Lite XML Document* as defined in the ContentDirectory service specification. (Care must be taken to correctly define namespaces.)

The *DIDL-Lite XML Document* describes a device-dependent (sub)set of imported properties (metadata) of the ContentDirectory service object that is referenced by the linked *xxxCDSObjectID* property. The information contained in the *DIDL-Lite XML Document* MUST exactly match the *DIDL-Lite XML Document* that would be returned in the *Result* argument of the *ContentDirectory::Browse()* action with its input arguments set as follows:

- **ObjectId**: The linked *xxxCDSObjectID* property value.
- **BrowseFlag**: Set to “*BrowseMetaData*”.
- **Filter**: Set to the list of property names that are imported from the ContentDirectory service by the ScheduledRecording service.
- **StartingIndex**: 0.
- **RequestedCount**: 0.
- **SortCriteria**: “”, the empty string.

The following example illustrates the possible content of a *cdsReference* property in the context of a *recordSchedule* object (expressed in XML).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<srs
 xmlns="urn:schemas-upnp-org:av:srs"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
 urn:schemas-upnp-org:av:srs
 http://www.upnp.org/schemas/av/srs.xsd"
 <item id="sched001">
   <class>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</class>
   <title>My Schedule</title>
   <scheduledCDSObjectID link="schedObj001">
     epg001
   </scheduledCDSObjectID>
   ...
 </item>

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

```xml
&lt;?xml version="1.0" encoding="UTF-8"?&gt;
&lt;DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"&gt;&lt;
 &lt;item id="epg001" parentID="container007"
   restricted="0"&gt;
   &lt;dc:title&gt;Friends&lt;/dc:title&gt;
   &lt;upnp:class&gt;
     object.item.epgItem.videoProgram
   &lt;/upnp:class&gt;
   ...
 &lt;/item&gt;
```

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
&lt;/DIDL-Lite&gt;

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

</cdsReference>

...  

</item>

</srs>
The next example illustrates the possible content of two `cdsReference` property instances relating to the `taskCDSObjectID` and `recordedCDSObjectID` property in the context of a `recordTask` object (expressed in XML).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<srs
 xmlns="urn:schemas-upnp-org:av:srs"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
 urn:schemas-upnp-org:av:srs
 http://www.upnp.org/schemas/av/srs.xsd">
 <item id="Task001">
   <class>OBJECT.RECORDTASK</class>
   <title>My Task</title>
   ...
 </item>
</srs>
```

```xml
<taskCDSObjectID link="tskObj001">
   epg001
</taskCDSObjectID>
```

```xml
<recordedCDSObjectID link="recObj001">
   rec001
</recordedCDSObjectID>
```

```xml
<cdsReference link="tskObj001">
<!--
The following DIDL-Lite XML Document needs to be interpreted as a simple string and therefore needs to be properly escaped
-->
   <![CDATA[
     &lt;?xml version="1.0" encoding="UTF-8"?&gt;
     &lt;?DIDL-Lite
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
     xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"&gt;
     &lt;item id="epg001" parentID="container007"
       restricted="0"&gt;
       &lt;dc:title&gt;Friends&lt;/dc:title&gt;
       &lt;upnp:class&gt;
         object.item.epgItem.videoProgram
       &lt;/upnp:class&gt;
       ...
     &lt;/item&gt;
     &lt;/DIDL-Lite&gt;
   ]]>
</cdsReference>
```

```xml
<cdsReference link="recObj001">
<!--
The following DIDL-Lite XML Document needs to be interpreted as a simple string and therefore needs to be properly escaped
-->
   <![CDATA[
     &lt;?xml version="1.0" encoding="UTF-8"?&gt;
     &lt;?DIDL-Lite
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
   ]]>
</cdsReference>
```
xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
&amp;lt;?item id="rec001" parentID="cnt009" restricted="0"&amp;gt;
&amp;lt;?dc:title&gt;My Show&amp;lt;/dc:title&gt;
&amp;lt;?upnp:class&gt;
object.item.videoItem
&amp;lt;/upnp:class&amp;gt;
…
&amp;lt;/item&amp;gt;
&amp;lt;?/DIDL-Lite&amp;gt;

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

</item>
</srs>
Appendix C. AV Working Committee Class Definitions (Normative)

C.1 Class Hierarchy

The ScheduledRecording service exposes a class hierarchy which is used to type all objects that can be retrieved from it. Each class is named using a string of the form described in Appendix D.3, “Class Name Syntax” below.

For a particular class, some properties are REQUIRED, others are OPTIONAL and some are PROHIBITED.

A class that is derived from another class MUST include all of the member properties of the parent class. The definition of a derived class MAY make some optional properties of the base class REQUIRED.

Each class definition includes a list of properties. Each property is expressed in XML as either an XML Element or an XML Attribute. Some independent properties are multi-valued for a class, meaning that, in an XML instance of the class, the property may occur more than once.

This Appendix defines the base class object from which all other classes are derived. No object of this abstract class can be instantiated. From the object class, two classes are derived; the object.recordSchedule class and the object.recordTask class.

The abstract object.recordSchedule class and its two derived abstract classes object.recordSchedule.direct and object.recordSchedule.query make up the basic hierarchy from which all other recordSchedule classes are derived. These three classes can not be instantiated (no object can exist within the ScheduledRecording service that has its class property set to “OBJECT.RECORDSCHEDULE”, “OBJECT.RECORDSCHEDULE.DIRECT” or “OBJECT.RECORDSCHEDULE.QUERY”).

The object.recordTask class is used to type all recordTask objects in the ScheduledRecording service. The object.recordTask class has no derived classes defined yet.

In addition to these classes, a number of classes are derived from the object.recordSchedule.direct and object.recordSchedule.query classes. Figure 7 below shows the hierarchy of these classes.

![Figure 7: Class hierarchy for the ScheduledRecording service.](image)

Vendors MAY extend the functionality, provided by the standard record classes, by adding vendor-defined properties. Any device that adds a property whose description matches one of the AV Working Committee-defined property descriptions MUST use the AV Working Committee-defined property name. In addition,
any device that uses a property name from the ScheduledRecording service specification MUST use it with
the same semantics as the AV Working Committee-defined description of that property. In order to
accommodate vendor-defined properties, control points should \textit{gracefully ignore} any properties whose
names and semantics they do not understand.

When adding properties, it is \textit{RECOMMENDED} that vendors create a vendor-defined derived class with a
vendor-defined class name, rather than adding the properties to the existing standard class without creating
a vendor-defined class. This provides a simple mechanism for control points to determine if a class has
been extended by simply examining the \texttt{class} property value. In all cases, vendor-defined classes MUST
remain fully compatible with the standard class from which they were derived. In other words, control
points that do not understand the specifics of the vendor-defined additions should still be able to interact
with an instance of the vendor-defined derived class object as if it were an instance of that standard class.

Vendor-defined classes MUST always be derived from standard classes that can be instantiated (the green-
colored boxes in Figure 7). It is therefore \textit{PROHIBITED} to derive vendor-defined classes directly from
classes, such as “\texttt{OBJECT.RECORDSCHEDULE}”, “\texttt{OBJECT.RECORDSCHEDULE.DIRECT}”, and
“\texttt{OBJECT.RECORDSCHEDULE.QUERY}”. It is allowed to derive vendor-defined classes from class
“\texttt{OBJECT.RECORDTASK}”.

All standard classes and vendor-defined derived classes supported by a particular ScheduledRecording
service implementation MUST be individually listed in the \texttt{allowedValueList} of the \texttt{class} property. (This
list can be retrieved via the \texttt{GetAllowedValues()} action.) Implementations are \textit{REQUIRED} to support all
intermediate classes in a chain of derived classes. For example, if an implementation supports a vendor-
defined class “\texttt{OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG.HDTV.LOCAL}”, then it MUST also
support the “\texttt{OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG.HDTV}” and
“\texttt{OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG}” classes.

As indicated in Appendix D.3, “Class Name Syntax”, strict naming conventions MUST be followed when
naming derived vendor-defined classes. Vendor-defined class names MUST be based on one of the
instantiatable class names in this specification. Therefore, all vendor-defined class names MUST start with
one of the string values, defined in Table B-2, “\texttt{allowedValueList} for the \texttt{class} Property”.

Control points that need to examine \texttt{class} property values, should be prepared to handle vendor-defined
class names. Therefore, control points should never use simple string matching to determine the actual
class of an object. Rather, they should parse the class name from left to right and determine if there is at
least a left substring match with one of the string values defined in Table B-2, “\texttt{allowedValueList} for the
\texttt{class} Property”. If such a match is found, the remaining characters in the class name can be examined for
potential matches with vendor-defined class names of which the control point is aware. If no such match is
found, the control point can treat the object as if it were an instance of the most specialized class for which
a match was found.

\textbf{C.1.1 Relationships between Classes and Properties}

The following tables present a complete overview of all the defined properties and in which classes these
properties are actually used (member properties).

For a particular class, some properties are \textit{REQUIRED}, others are \textit{OPTIONAL} and some are
\textit{PROHIBITED}. Every instance of a class MUST have a value for each supported \textit{REQUIRED} or
\textit{OPTIONAL} member property of that class (see Section 2.2, “Terms”).

The support level of a member property defines how the member property MUST be used in the arguments
of an action when that action is invoked. The support level of a member property can be different for
\texttt{recordSchedule}, \texttt{recordScheduleParts}, and \texttt{recordTask} usage.

The \texttt{recordScheduleParts} support level for the specified class indicates the use of a member property when
a control point requests to create a \texttt{recordSchedule}. If a member property is defined as \textit{REQUIRED} for
\texttt{recordScheduleParts} usage, an argument of type \texttt{A_ARG_TYPE_RecordScheduleParts} MUST contain that
member property and the ScheduledRecording service MUST support it. If it is defined as \textit{OPTIONAL}, the
ScheduledRecording service MAY support the member property and a control point may specify or omit

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
the member property in a request message even if the member property is supported by the ScheduledRecording service. PROHIBITED or unsupported OPTIONAL member properties specified in an argument of type A_ARG_TYPE_RecordScheduleParts MUST be gracefully ignored by the ScheduledRecording service. The set of properties that are supported for an argument of type A_ARG_TYPE_RecordScheduleParts can be retrieved by specifying “A_ARG_TYPE_RecordScheduleParts” in the DataTypeID argument when invoking the GetPropertyList() action. The support level for each of those supported member properties of each class can be retrieved by invoking the GetAllowedValues() action.

The recordSchedule support level for the specified class indicates the use of a member property when a control point retrieves a recordSchedule object. If a member property is defined as REQUIRED for recordSchedule usage, an argument of type A_ARG_TYPE_RecordSchedule MUST contain that member property and the ScheduledRecording service MUST support it. OPTIONAL supported member properties that are enumerated in the Filter argument MUST also be specified in the argument. If the resulting XML is not a valid document, other OPTIONAL properties MUST be added to create the smallest valid XML document. If the action does not have a Filter argument (like the CreateRecordSchedule() action), the action MUST return all OPTIONAL supported member properties (as if the Filter argument were set to "*:*"). If a control point does not specify a supported OPTIONAL member property in a request, the ScheduledRecording service MUST add it into the response and provide its default setting. The set of properties that are supported for an argument of type A_ARG_TYPE_RecordSchedule can be retrieved by specifying “A_ARG_TYPE_RecordSchedule” in the DataTypeID argument when invoking the GetPropertyList() action. The support level for each of those supported member properties of each class can be retrieved by invoking the GetAllowedValues() action.

The recordTask support level for the specified class indicates the use of a member property when a control point retrieves a recordTask object. If a member property is defined as REQUIRED for recordTask usage, an argument of type A_ARG_TYPE_RecordTask MUST contain that member property and the ScheduledRecording service MUST support it. OPTIONAL supported member properties that are enumerated in the Filter argument MUST also be specified in the argument. If the resulting XML is not a valid document, other OPTIONAL properties MUST be added to create the smallest valid XML document. The set of properties that are supported for an argument of type A_ARG_TYPE_RecordTask can be retrieved by specifying “A_ARG_TYPE_RecordTask” in the DataTypeID argument when invoking the GetPropertyList() action. The support level for each of those supported member properties of each class can be retrieved by invoking the GetAllowedValues() action.

Dependent properties are PROHIBITED if their associated independent property does not exist. They can be REQUIRED or OPTIONAL when the independent property does exist.

### C.1.2 recordScheduleParts Properties

The following table indicates the support level (REQUIRED, OPTIONAL, PROHIBITED, or UNDEFINED) of a property when used in an argument of type A_ARG_TYPE_RecordScheduleParts for each class. The \( \sqrt{\text{\textbackslash n}} \) mark indicates that the property’s support level is inherited from the parent class. The coloring still indicates the support level.
## Table C-1: Class Properties Overview for `recordScheduleParts` usage

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Common Properties</th>
<th>Base Properties</th>
<th>Priority Properties</th>
<th>Output Control Properties</th>
<th>Schedule Only Properties</th>
<th>UNDEFINED</th>
<th>INHERITED</th>
<th>PROHIBITED</th>
<th>OPTIONAL</th>
<th>REQUIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>@id</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>title</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cdsReference</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cdsReference@link</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredPriority</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkNum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledStartDateDateTime</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledDuration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledProgramCode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Property Name

<table>
<thead>
<tr>
<th>Property Name</th>
<th>object</th>
<th>recordSchedule</th>
<th>direct</th>
<th>manual</th>
<th>cdsEPG</th>
<th>cdsNonEPG</th>
<th>programCode</th>
<th>query</th>
<th>contentName</th>
<th>contentID</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduledProgramCode@type</td>
<td>L</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

### Matching Content Criteria

| matchingName                          | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| matchingName@type                     | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| matchingName@substringMatch           | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| matchingID                            | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| matchingID@type                       | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |

### Matching Qualifying Criteria

| matchingChannelID                     | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| matchingChannelID@type                | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| matchingChannelID@distriNetworkName   | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| matchingChannelID@distriNetworkID     | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| matchingStartDateDateTimeRange        | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| matchingDurationRange                 | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| matchingRatingLimit                   | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| matchingRatingLimit@type              | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |

### Content Control Properties

| totalDesiredRecordTasks               | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| scheduledStartDateDateTimeAdjust      | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| scheduledDurationAdjust               | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| activePeriod                          | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| durationLimit                         | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| durationLimit@effect                   | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| channelMigration                      | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| timeMigration                         | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| allowDuplicates                       | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |

### Storage Related Properties

<p>| persistedRecordings                   | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| persistedRecordings@latest            | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| persistedRecordings@preAllocation     | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |
| persistedRecordings@storedLifetime    | L      | O              | O      | O      | O      | O         | O           | O     | O           | O         |</p>
<table>
<thead>
<tr>
<th>Property Name</th>
<th>Schedule State Properties</th>
<th>Statistics Properties</th>
<th>Task Only Properties</th>
<th>General Properties</th>
<th>Content ID Related Properties</th>
<th>Matched Content Criteria</th>
<th>Matched Qualifying Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleState</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>scheduleState@currentErrors</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>abnormalTasksExist</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>currentRecordTaskCount</td>
<td>OPTIONAL</td>
<td>OPTIONAL</td>
<td>OPTIONAL</td>
<td>OPTIONAL</td>
<td>OPTIONAL</td>
<td>OPTIONAL</td>
<td>OPTIONAL</td>
</tr>
<tr>
<td>totalCreatedRecordTasks</td>
<td>PROHIBITED</td>
<td>PROHIBITED</td>
<td>PROHIBITED</td>
<td>PROHIBITED</td>
<td>PROHIBITED</td>
<td>PROHIBITED</td>
<td>PROHIBITED</td>
</tr>
<tr>
<td>totalCompletedRecordTasks</td>
<td>UNDEFINED</td>
<td>UNDEFINED</td>
<td>UNDEFINED</td>
<td>UNDEFINED</td>
<td>UNDEFINED</td>
<td>UNDEFINED</td>
<td>UNDEFINED</td>
</tr>
<tr>
<td>recordScheduleID</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>recordedCDSObjectID</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>recordedCDSObjectID@link</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>taskCDSObjectID</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>taskCDSObjectID@link</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>taskChannelID</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>taskChannelID@type</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>taskChannelID@distriNetworkName</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>taskChannelID@distriNetworkID</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>taskStartDateTime</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>taskDuration</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>taskProgramCode</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>taskProgramCode@type</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>recordQuality</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>recordQuality@type</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>matchedName</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>matchedName@type</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>matchedID</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
<tr>
<td>matchedID@type</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
<td>REQUIRED</td>
</tr>
</tbody>
</table>
### recordSchedule Properties

The next table indicates the support level (REQUIRED, OPTIONAL, PROHIBITED or UNDEFINED) of a property when used in an argument of type `A_ARG_TYPE_RecordSchedule` for each class. The √ mark indicates that the property’s support level is inherited from the parent class. The marking still indicates the support level.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>object</th>
<th>recordSchedule</th>
<th>direct</th>
<th>manual</th>
<th>codeEPG</th>
<th>codeNonEPG</th>
<th>contentName</th>
<th>contentID</th>
</tr>
</thead>
<tbody>
<tr>
<td>matchedRating</td>
<td>U</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedRating@type</td>
<td>P</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedEpisodeType</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content Control Properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskStartDateTimeAdjust</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskDurationAdjust</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskDurationLimit</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskDurationLimit@effect</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskChannelMigration</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskTimeMigration</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task State Properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@phase</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@startDateTimeMet</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@endDateTimeMet</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@recording</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@someBitsRecorded</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@someBitsMissing</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@firstBitsRecorded</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@lastBitsRecorded</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@fatalError</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@currentErrors</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@errorHistory</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@pendingErrors</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskState@infoList</td>
<td>L</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table C-2: Class Properties Overview for `recordSchedule` usage

<table>
<thead>
<tr>
<th>Property Name</th>
<th>object</th>
<th>recordSchedule</th>
<th>direct</th>
<th>manual</th>
<th>cds-EPG</th>
<th>cds-NonEPG</th>
<th>programCode</th>
<th>query</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Common Properties</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@id</td>
<td>R</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>R</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>R</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cdsReference</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cdsReference@link</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Priority Properties</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredPriority</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Output Control Properties</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredrecordQuality</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredrecordQuality@type</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Schedule Only Properties</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkNum</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkID</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledStartDateTime</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledDuration</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledProgramCode</td>
<td>U</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Property Name

<table>
<thead>
<tr>
<th>Property Name</th>
<th>object</th>
<th>recordSchedule</th>
<th>direct</th>
<th>manual</th>
<th>EPG</th>
<th>nonEPG</th>
<th>contentName</th>
<th>contentID</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scheduledProgramCode@type</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><code>matchingName</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><code>matchingName@type</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><code>matchingName@substringMatch</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><code>matchingID</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><code>matchingID@type</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><code>Matching Qualifying Criteria</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>matchingChannelID</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td><code>matchingChannelID@type</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td><code>matchingChannelID@distriNetworkName</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td><code>matchingChannelID@distriNetworkID</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>O</td>
<td>P</td>
</tr>
<tr>
<td><code>matchingStartDateDateTimeRange</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td><code>matchingDurationRange</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td><code>matchingRatingLimit</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td><code>matchingRatingLimit@type</code></td>
<td>L</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td><code>Matching Content Criteria</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>scheduledStartDateTimeAdjust</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td><code>scheduledDurationAdjust</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td><code>activePeriod</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td><code>durationLimit</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td><code>durationLimit@effect</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td><code>channelMigration</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td><code>timeMigration</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td><code>allowDuplicates</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td><code>Storage Related Properties</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>persistedRecordings</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td><code>persistedRecordings@latest</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td><code>persistedRecordings@preAllocation</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td><code>persistedRecordings@storedLifetime</code></td>
<td>L</td>
<td>O</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>Property Name</td>
<td>Schedule State Properties</td>
<td>Statistics Properties</td>
<td>Task Only Properties</td>
<td>General Properties</td>
<td>Content ID Related Properties</td>
<td>Matched Content Criteria</td>
<td>Matched Qualifying Criteria</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------</td>
<td>----------------------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>-----------------------------</td>
<td>--------------------------</td>
<td>---------------------------</td>
<td></td>
</tr>
<tr>
<td>scheduleState</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduleState@currentErrors</td>
<td>OPTIONAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abnormalTasksExist</td>
<td>PROHIBITED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>currentRecordTaskCount</td>
<td>UNDEFINED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>totalCreatedRecordTasks</td>
<td>UNDEFINED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>totalCompletedRecordTasks</td>
<td>UNDEFINED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordScheduleID</td>
<td>INHERITED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordedCDSObjectID</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordedCDSObjectID@link</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskCDSObjectID</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskCDSObjectID@link</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskChannelID</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskChannelID@type</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskChannelID@distriNetworkName</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskChannelID@distriNetworkID</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskStartDateTime</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskDuration</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskProgramCode</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskProgramCode@type</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordQuality</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordQuality@type</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedName</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedName@type</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedID</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedID@type</td>
<td>REQUIRED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Property Name

<table>
<thead>
<tr>
<th>Property Name</th>
<th>object</th>
<th>recordSchedule</th>
<th>direct</th>
<th>manual</th>
<th>codeEPG</th>
<th>codeNonEPG</th>
<th>StartTime</th>
<th>EndTime</th>
<th>recording</th>
<th>someBitsRecorded</th>
<th>someBitsMissing</th>
<th>firstBitsRecorded</th>
<th>lastBitsRecorded</th>
<th>fatalError</th>
<th>currentErrors</th>
<th>errorHistory</th>
<th>pendingErrors</th>
<th>infoList</th>
</tr>
</thead>
</table>

### C.1.4 recordTask Properties

The next table indicates the support level (REQUIRED, OPTIONAL, PROHIBITED or UNDEFINED) of a property when used in an argument of type `A_ARG_TYPE_RecordTask`. The √ mark indicates that the property’s support level is inherited from the parent class. The coloring still indicates the support level.
### Table C-3: Class Properties Overview for `recordTask` usage

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Common Properties</th>
<th>Base Properties</th>
<th>Priority Properties</th>
<th>Output Control Properties</th>
<th>Schedule Only Properties</th>
<th>Content ID Related Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>@id</td>
<td>title</td>
<td>recordDestination</td>
<td>scheduledCDSObjectID</td>
<td>scheduledCDSObjectID@link</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scheduledCDSObjectID@link</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scheduledChannelID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scheduledChannelID@link</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scheduledChannelID@type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scheduledChannelID@distriNetworkName</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scheduledChannelID@distriNetworkID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scheduledStartDateTime</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scheduledDuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scheduledProgramCode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>scheduledProgramCode@type</td>
</tr>
<tr>
<td></td>
<td>REQUIRED</td>
<td>PROHIBITED</td>
<td>UNDEFINED</td>
<td>INHERITED</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
<table>
<thead>
<tr>
<th>Property Name</th>
<th>object</th>
<th>recordTask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matching Content Criteria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingName@type</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingName@subStringMatch</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingID</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingID@type</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>Matching Qualifying Criteria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingChannelID</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingChannelID@type</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkName</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkID</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingStartDateTimeRange</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingDurationRange</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingRatingLimit</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingRatingLimit@type</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>matchingEpisodeType</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>Content Control Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>totalDesiredRecordTasks</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>scheduledStartDateTimeAdjust</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>scheduledDurationAdjust</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>activePeriod</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>durationLimit</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>durationLimit@effect</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>channelMigration</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>timeMigration</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>allowDuplicates</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>Storage Related Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>persistedRecordings</td>
<td>E</td>
<td>P</td>
</tr>
<tr>
<td>persistedRecordings@latest</td>
<td>E</td>
<td>P</td>
</tr>
<tr>
<td>persistedRecordings@preAllocation</td>
<td>E</td>
<td>P</td>
</tr>
<tr>
<td>persistedRecordings@storedLifetime</td>
<td>E</td>
<td>P</td>
</tr>
<tr>
<td>Schedule State Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduleState</td>
<td>E</td>
<td>P</td>
</tr>
</tbody>
</table>

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
<table>
<thead>
<tr>
<th>Property Name</th>
<th>R</th>
<th>O</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleState@currentErrors</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abnormalTasksExist</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistics Properties</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>currentRecordTaskCount</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>totalCreatedRecordTasks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>totalCompletedRecordTasks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task Only Properties</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General Properties</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordScheduleID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordedCDSObjectID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordedCDSObjectID@link</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content ID Related Properties</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskCDSObjectID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskCDSObjectID@link</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskChannelID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskChannelID@type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskChannelID@distriNetworkName</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskChannelID@distriNetworkID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskStartDateTime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskDuration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskProgramCode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>taskProgramCode@type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordQuality</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordQuality@type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matched Content Criteria</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedName</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedName@type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedID@type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matched Qualifying Criteria</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedRating</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedRating@type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchedEpisodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The following sections define the standard record classes. The support level of the available properties for each class is also indicated. Vendors MAY add vendor-dependent properties to any of the defined classes. An instance of a normative class MUST NOT add properties, other than the properties already listed for each class definition below. In other words, a given instance of a record class can only have:

- The properties listed for that class (as per each definition below).
- The properties that are defined members of the parent class.
- Vendor-defined properties that are using other XML namespace(s).
- ContentDirectory service properties imported with a normative namespace prefix (see Appendix B.17, “ContentDirectory Service Imported Properties”).

C.2 Class Definitions
C.3 **object** Base Class

This is the abstract base class for the entire ScheduledRecording service class hierarchy. No object of this abstract class can be instantiated. The **object** class defines properties that are common to all ScheduledRecording service objects. The table below lists all standard defined properties (see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for each property.

### Table C-4: **object** Base Class Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Property Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base Properties</strong></td>
<td><strong>Content Control Properties</strong></td>
</tr>
<tr>
<td>@id</td>
<td>totalDesiredRecordTasks</td>
</tr>
<tr>
<td>title</td>
<td>scheduledStartDateTimeAdjust</td>
</tr>
<tr>
<td>class</td>
<td>scheduledDurationAdjust</td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td>activePeriod</td>
</tr>
<tr>
<td>cdsReference</td>
<td>durationLimit</td>
</tr>
<tr>
<td>cdsReference@link</td>
<td>channelMigration</td>
</tr>
<tr>
<td>Priority Properties</td>
<td>timeMigration</td>
</tr>
<tr>
<td>priority</td>
<td>allowDuplicates</td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td></td>
</tr>
<tr>
<td>desiredPriority</td>
<td></td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td></td>
</tr>
<tr>
<td>Output Control Properties</td>
<td></td>
</tr>
<tr>
<td>recordDestination</td>
<td></td>
</tr>
<tr>
<td>recordDestination@typeName</td>
<td></td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td></td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td></td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td></td>
</tr>
<tr>
<td>Content ID Related Properties</td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkName</td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkID</td>
<td></td>
</tr>
<tr>
<td>scheduledStartDateTime</td>
<td></td>
</tr>
<tr>
<td>scheduledDuration</td>
<td></td>
</tr>
<tr>
<td>Scheduled Program Code</td>
<td></td>
</tr>
<tr>
<td>scheduledProgramCode@type</td>
<td></td>
</tr>
<tr>
<td>matching Content Criteria</td>
<td></td>
</tr>
<tr>
<td>matchingName</td>
<td></td>
</tr>
<tr>
<td>matchingName@type</td>
<td></td>
</tr>
<tr>
<td>matchingName@subStringMatch</td>
<td></td>
</tr>
<tr>
<td>matchingID</td>
<td></td>
</tr>
<tr>
<td>matchingID@type</td>
<td></td>
</tr>
<tr>
<td>Matching Qualifying Criteria</td>
<td></td>
</tr>
<tr>
<td>matchingChannelID@type</td>
<td></td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkName</td>
<td></td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkID</td>
<td></td>
</tr>
<tr>
<td>matchingStartDateTimeRange</td>
<td></td>
</tr>
<tr>
<td>matchingStartDateTimeRange</td>
<td></td>
</tr>
<tr>
<td>matchingDurationRange</td>
<td></td>
</tr>
<tr>
<td>matchingRatingLimit</td>
<td></td>
</tr>
<tr>
<td>matchingRatingLimit@type</td>
<td></td>
</tr>
<tr>
<td>matchingEpisodeType</td>
<td></td>
</tr>
<tr>
<td>Storage Related Properties</td>
<td></td>
</tr>
<tr>
<td>persistedRecordings</td>
<td></td>
</tr>
<tr>
<td>persistedRecordings@latest</td>
<td></td>
</tr>
<tr>
<td>persistedRecordings@preAllocation</td>
<td></td>
</tr>
<tr>
<td>persistedRecordings@storedLifetime</td>
<td></td>
</tr>
<tr>
<td>Schedule State Properties</td>
<td></td>
</tr>
<tr>
<td>scheduleState</td>
<td></td>
</tr>
<tr>
<td>scheduleState@currentErrors</td>
<td></td>
</tr>
<tr>
<td>abnormalTasksExist</td>
<td></td>
</tr>
<tr>
<td>Statistics Properties</td>
<td></td>
</tr>
<tr>
<td>currentRecordTaskCount</td>
<td></td>
</tr>
<tr>
<td>totalCreatedRecordTasks</td>
<td></td>
</tr>
<tr>
<td>totalCompletedRecordTasks</td>
<td></td>
</tr>
<tr>
<td>Task General Properties</td>
<td></td>
</tr>
<tr>
<td>recordScheduledID</td>
<td></td>
</tr>
<tr>
<td>recordedCDSObjectID@link</td>
<td></td>
</tr>
<tr>
<td>recordedCDSObjectID@type</td>
<td></td>
</tr>
<tr>
<td>Task Content ID Properties</td>
<td></td>
</tr>
<tr>
<td>taskCDSObjectID</td>
<td></td>
</tr>
<tr>
<td>taskCDSObjectID@link</td>
<td></td>
</tr>
<tr>
<td>taskChannelID@type</td>
<td></td>
</tr>
<tr>
<td>taskChannelID@distriNetworkName</td>
<td></td>
</tr>
<tr>
<td>taskChannelID@distriNetworkID</td>
<td></td>
</tr>
<tr>
<td>taskStartDateTime</td>
<td></td>
</tr>
<tr>
<td>taskDuration</td>
<td></td>
</tr>
<tr>
<td>taskProgramCode@type</td>
<td></td>
</tr>
<tr>
<td>recordQuality</td>
<td></td>
</tr>
<tr>
<td>recordQuality@type</td>
<td></td>
</tr>
<tr>
<td>Matched Content Criteria</td>
<td></td>
</tr>
<tr>
<td>matchedName</td>
<td></td>
</tr>
<tr>
<td>matchedName@type</td>
<td></td>
</tr>
<tr>
<td>matchedID</td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
C.3.1 object.recordSchedule Class

This is the abstract base class for the ScheduledRecording service record schedules class hierarchy. No object of this abstract class can be instantiated. The object.recordSchedule class defines properties that are common to all object.recordSchedule list entries. The table below lists all recordSchedule-related standard defined properties (recordTask-only properties are omitted from the table – see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for recordScheduleParts (RSP) and recordSchedule (RS) usage for each property.

Table C-5: object.recordSchedule Base Class Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@id</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>title</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>class</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td>P</td>
<td>O</td>
</tr>
<tr>
<td>cdsReference</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>cdsReference@link</td>
<td>P</td>
<td>O</td>
</tr>
<tr>
<td>Priority Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td>P</td>
<td>O</td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>desiredPriority</td>
<td>P</td>
<td>O</td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Output Control Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination</td>
<td>D</td>
<td>L</td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td>D</td>
<td>L</td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td>D</td>
<td>O</td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td>D</td>
<td>P</td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td>D</td>
<td>O</td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Content ID Related Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkName</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledStartDate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matching Content Criteria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName@substringMatch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingID@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matching Qualifying Criteria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingChannelID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingChannelID@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkName</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingStartDateRange</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingDurationRange</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingRatingLimit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingRatingLimit@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingEpisodeType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content Control Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>totalDesiredRecordTasks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledStartDateAdjust</td>
<td>D</td>
<td>O</td>
</tr>
<tr>
<td>scheduledDurationAdjust</td>
<td>D</td>
<td>O</td>
</tr>
</tbody>
</table>

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
C.3.1.1 **object.recordSchedule.direct** Class

The `object.recordSchedule.direct` abstract class is derived from the `object.recordSchedule` class. No object of this abstract class can be instantiated.

The main characteristic of the `object.recordSchedule.direct` class is that all the information that is needed to create associated `recordTask` instances is contained within the properties of the `recordSchedule`. The properties contain sufficient information to allow the ScheduledRecording service to translate this information into a deterministic set of `recordTask` properties. For example, if a ScheduledRecording service implementation supports the `object.recordSchedule.direct.programCode` class, the ScheduledRecording service is able to interpret the `scheduledProgramCode` property and derive the appropriate `taskStartDate`, `taskStartTime`, `taskDuration`, and `taskChannelID` `recordTask` properties from it.

The table below lists all standard defined properties (see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for `recordScheduleParts` (RSP) and `recordSchedule` (RS) usage for each property.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base Properties</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@id</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>title</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>class</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>cdsReference</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>cdsReferenceRef</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td><strong>Priority Properties</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td>√</td>
<td>X</td>
</tr>
<tr>
<td>desiredPriority</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td><strong>Output Control Properties</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td><strong>Content ID Related Properties</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table C-6: **object.recordSchedule.direct** Class Properties
### C.3.1.1.1 object.recordSchedule.direct.manual Class

The `object.recordSchedule.direct.manual` class is used to create `recordSchedule` instances for manual scheduling of recordings. The content to be recorded is uniquely identified by the `scheduledChannelID`, `scheduledStartDateTime`, and `scheduledDuration` properties.

The table below lists all standard defined properties (see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for `recordScheduleParts` (RSP) and `recordSchedule` (RS) usage for each property.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>@id</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>title</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>class</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>cdsReference</td>
<td>√</td>
<td>P</td>
</tr>
<tr>
<td>cdsReference@link</td>
<td>√</td>
<td>P</td>
</tr>
<tr>
<td>priority</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredPriority</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td>√</td>
<td>P</td>
</tr>
<tr>
<td>scheduledCDSObjectID@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkID</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkName</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>scheduledStartDateTime</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>scheduledDuration</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>scheduledProgramCode</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledProgramCode@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingName</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingName@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingName@substringMatch</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingID</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingID@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingChannelID</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingChannelID@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkName</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkID</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>matchingStartDateTimeRange</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingDurationRange</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingRatingLimit</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingRatingLimit@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingProgramCode</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingProgramCode@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingEpisodeType</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
C.3.1.1.2  object.recordSchedule.direct.cdsEPG Class

The object.recordSchedule.direct.cdsEPG class is used to create recordSchedule instances for scheduling of recordings, based on local EPG information. The content to be recorded is uniquely identified by the scheduledCDSObjectID property that MUST reference an EPG item (object.item.epgItem class) in an associated ContentDirectory service. Most EPG item types currently defined identify only a single recording event. In the future, new EPG item types may be defined that identify multiple recording events.

The REQUIRED association between a ContentDirectory service and a ScheduledRecording service is established by having both services reside within the same UPnP MediaServer device. See also Appendix E, “ScheduledRecording Service Relationship to ContentDirectory Service” and Appendix F, “ScheduledRecording Service Relationship to EPG” for further details.

The table below lists all standard defined properties (see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for recordScheduleParts (RSP) and recordSchedule (RS) usage for each property.

Table C-8: object.recordSchedule.direct.cdsEPG Class Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@id</td>
<td></td>
<td></td>
</tr>
<tr>
<td>title</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cdsReference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cdsReference@link</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredPriority</td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Control Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content ID Related Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledStartDate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledDuration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledProgramCode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledProgramCode@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matching Content Criteria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName@caseSensitivity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName@enlStringMatch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName@pattern</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingID@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matching Qualifying Criteria</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Property Name
<table>
<thead>
<tr>
<th></th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>matchingChannelID</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>matchingChannelId@type</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>matchingChannelId@distriNetworkName</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>matchingChannelId@distriNetworkID</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>matchingStartDateRange</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>matchingDurationRange</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>matchingRatingLimit</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>matchingRatingLimit@type</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>matchingEpisodeType</td>
<td>X</td>
<td>O</td>
</tr>
</tbody>
</table>

### Content Control Properties
- `totalDesiredRecordTasks`: X, O
- `scheduledStartDateAdjust`: O, X
- `scheduledDurationAdjust`: O, X
- `activePeriod`: O, O
- `durationLimit`: O, O
- `durationLimit@effect`: O, O
- `channelMigration`: O, O

### Storage Related Properties
- `persistedRecordings`: X, O
- `persistedRecordings@latest`: X, O
- `persistedRecordings@preAllocation`: X, O
- `persistedRecordings@storedLifetime`: X, O

### Schedule State Properties
- `scheduleState`: O, X
- `scheduleState@currentErrors`: O, X
- `abnormalTasksExist`: O, X

### Statistics Properties
- `currentRecordTaskCount`: X, X
- `totalCreatedRecordTasks`: X, X
- `totalCompletedRecordTasks`: X, X

## C.3.1.1.3 `object.recordSchedule.direct.cdsNonEPG` Class
The `object.recordSchedule.direct.cdsNonEPG` class is used to create `recordSchedule` instances for scheduling of recordings, for which (only) channel information is available in a local ContentDirectory database. The content to be recorded is uniquely identified by the `scheduledStartDate` and `scheduledDuration` properties, supplemented with the `scheduledCDSObjectID` property that MUST reference a ContentDirectory service object whose class is not “`object.item.epgItem`” or derived from that class. Additionally, the referenced ContentDirectory service object MUST identify content that will be available for recording at the time the recording is scheduled to start.

Examples of applicable ContentDirectory service objects are:
- A User Channel object that contains specific channel information.
- An object that represents an analog A/V input connection to the device.
- An object that represents an IP network program feed.
- An object that represents an already existing file.
- Etc.

The REQUIRED association between a ContentDirectory service and a ScheduledRecording service is established by having both services reside within the same UPnP MediaServer device. See also Appendix E, “ScheduledRecording Service Relationship to ContentDirectory Service” and Appendix F, “ScheduledRecording Service Relationship to EPG” for further details.
The table below lists all standard defined properties (see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for recordScheduleParts (RSP) and recordSchedule (RS) usage for each property.

Table C-9: object.recordSchedule.direct.cdsNonEPG Class Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@id</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>title</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>class</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>additionalStatusInfos</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>cdsReference</td>
<td>√</td>
<td>O</td>
</tr>
<tr>
<td>cdsReference@link</td>
<td>√</td>
<td>R</td>
</tr>
<tr>
<td>Priority Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredPriority</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Output Control Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content ID Related Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkName</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkID</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledStartDate</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>scheduledDuration</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>scheduledProgramCode</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledProgramCode@type</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Matching Content Criteria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName@substringMatch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingID@type</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C.3.1.1.4 object.recordSchedule.direct.programCode Class

The object.recordSchedule.direct.programCode class is used to create recordSchedule instances for scheduling of recordings, based on program code information. The content to be recorded is uniquely identified by the scheduledProgramCode property that contains a unique code that can be translated by the ScheduledRecording service into a precise start date, start time, duration and channel for the recording event(s). However, most program code types currently defined identify only a single recording event. In the future, new program code types may be defined that identify multiple recording events.
The table below lists all standard defined properties (see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for recordScheduleParts (RSP) and recordSchedule (RS) usage for each property.

Table C-10: object.recordSchedule.direct.programCode Class Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>@id</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>title</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>class</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>cdsReference</td>
<td>√</td>
<td>P</td>
</tr>
<tr>
<td>cdsReference@link</td>
<td>√</td>
<td>P</td>
</tr>
<tr>
<td>priority</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredPriority</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td>√</td>
<td>P</td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkName</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkID</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledStartDate</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledDuration</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledProgramCode</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledProgramCode@type</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>matchingName</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingName@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingName@substringMatch</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingID</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingID@type</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Matching Qualifying Criteria

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>matchingChannelID</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingChannelID@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkName</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingChannelID@distriNetworkID</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingStartDateRange</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingDurationRange</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingRatingLimit</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingRatingLimit@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingEpisodeType</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Content Control Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalDesiredRecordTasks</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledDurationAdjust</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>activePeriod</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>durationLimit</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>durationLimit@effect</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>channelMigration</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>timeMigration</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>allowDuplicates</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Storage Related Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>persistedRecordings</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>persistedRecordings@latest</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>persistedRecordings@preAllocation</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>persistedRecordings@storedLifetime</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Schedule State Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>scheduleState</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduleState@currentErrors</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>abnormalTasksExist</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Statistics Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>currentCreatedRecordTasks</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>totalCreatedRecordTasks</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>totalCompletedRecordTasks</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

C.3.1.2 object.recordSchedule.query Class

The object.recordSchedule.query abstract class is derived from the recordSchedule base class. No object of this abstract class can be instantiated.

The main characteristic of the object.recordSchedule.query class is that the properties of the recordSchedule are used as matching criteria to select items from external sources (like EPG databases, side-band metadata streams in digital broadcasts, etc.). After appropriate searching and matching, the metadata from these external items is used to populate recordTask instances. This process ensures that the recordTask properties match the rules set forth in the recordSchedule’s properties (matching criteria).
The table below lists all standard defined properties (see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for recordScheduleParts (RSP) and recordSchedule (RS) usage for each property.

Table C-11: object.recordSchedule.query Class Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@id</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>title</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>class</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td>Y</td>
<td>P</td>
</tr>
<tr>
<td>cdsReference</td>
<td>Y</td>
<td>P</td>
</tr>
<tr>
<td>cdsReference@link</td>
<td>Y</td>
<td>P</td>
</tr>
<tr>
<td>Priority Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>desiredPriority</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Output Control Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td>Y</td>
<td>P</td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td>Y</td>
<td>P</td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Content ID Related Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkName</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkID</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledStartDateTime</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledDuration</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledProgramCode</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>scheduledProgramCode@type</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Matching Content Criteria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingName@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingName@substringMatch</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingID</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>matchingID@type</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

C.3.1.2.1 object.recordSchedule.query.contentName Class

The object.recordSchedule.query.contentName class is used to create recordSchedule instances for scheduling of recordings, based on program or series name information. The content to be recorded is determined by matching the value, specified in the matchingName property to the names of content items made available to the ScheduledRecording service by REQUIRED external resources like access to EPG databases, access to Service Information side-band data in digital broadcasts, etc. The matching process can be further restricted by providing a combination of Matching Qualifying Criteria properties. Any external content item MUST match those additional criteria to be considered a potential candidate for recording.
The table below lists all standard defined properties (see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for recordScheduleParts (RSP) and recordSchedule (RS) usage for each property.

Table C-12: object.recordSchedule.query.contentName Class Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@id</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>title</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cdsReference</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>cdsReference@link</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Priority Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredPriority</td>
<td></td>
<td></td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Control Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Content ID Related Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distribNetworkName</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduledChannelID@distribNetworkID</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduledStartDateTime</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduledDuration</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduledDurationAdjust</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>activePeriod</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>durationLimit</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>durationLimit@effect</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>channelMigration</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>timeMigration</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>allowDuplicates</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Content Control Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>totalDesiredRecordTasks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledStartDateAdjust</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledDurationAdjust</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistics Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>persistedRecordings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>persistedRecordings@latest</td>
<td></td>
<td></td>
</tr>
<tr>
<td>persistedRecordings@preAllocation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>persistedRecordings@storedLifetime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schedule State Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduleState</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>scheduleState@currentErrors</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>abnormalTasksExist</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistics Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>currentRecordTaskCount</td>
<td></td>
<td></td>
</tr>
<tr>
<td>totalCreatedRecordTasks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>totalCompletedRecordTasks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C.3.1.2.2 object.recordSchedule.query.contentID Class

The object recordSchedule.query.contentID class is used to create recordSchedule instances for scheduling of recordings, based on program or series ID information. The content to be recorded is determined by matching the value, specified in the matchingID property to the IDs of content items made available to the ScheduledRecording service by REQUIRED external resources like access to EPG databases, access Service Information side-band data in digital broadcasts, etc. The matching process can be further restricted by providing a combination of Matching Qualifying Criteria properties. Any external content item MUST match those additional criteria to be considered a potential candidate for recording.
The table below lists all standard defined properties (see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, REQUIRED, and UNDEFINED) in this class for both input and output for each property.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RSP</th>
<th>RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>@id</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>title</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>class</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>additionalStatusInfo</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>cdsReference</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>cdsReference@link</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Priority Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>priority@orderedValue</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredPriority</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredPriority@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Output Control Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordDestination</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@mediaType</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@targetURL</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>recordDestination@preference</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredRecordQuality</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>desiredRecordQuality@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Content ID Related Properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduledCDSObjectID</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledCDSObjectID@link</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledChannelID</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledChannelID@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkName</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledChannelID@distriNetworkID</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledDateTime</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledDuration</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledProgramCode</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>scheduledProgramCode@type</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Matching Content Criteria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>matchingName</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>matchingName@type</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>matchingName@subStringMatch</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>matchingID</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>matchingID@type</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Copyright © 2008, Contributing Members of the UPnP™ Forum. All rights Reserved.
C.3.2 **object.recordTask** Class

This is the base class for the ScheduledRecording service record task class hierarchy. Currently, this is the only class defined in this hierarchy. All `recordTask` objects in the ScheduledRecording service are members of this class. The `object.recordTask` class defines properties that are common to all `recordTask` list entries.

A `recordTask` object represents an actual recording occurrence. More sophisticated ScheduledRecording service implementations MAY implement OPTIONAL actions that allow a control point to manipulate individual `recordTask` instances. For example, the OPTIONAL `DisableRecordTask()` action can be used to selectively disable (that is: recording task suspended and any actual recording MUST NOT occur) one or more `recordTask` instances, spawned from the same `recordSchedule` if not all recordings are desired.

A `recordTask` SHOULD be created by the ScheduledRecording service as soon as all necessary information (like EPG data) becomes available. It SHOULD be maintained at least until the `recordTask` has finished. It is RECOMMENDED to maintain all completed `recordTask` instances for a reasonable time or until space is needed so that control points can retrieve `recordTask` state information after the recording has finished.

One or more `recordTask` instances can be created per `recordSchedule`. Some `recordSchedule` instances may not have a `recordTask` because they have not scheduled any recordings yet.

The list of the `recordTask` instances can be obtained using the `BrowseRecordTasks()` action. A `recordTask` can be disabled using the `DisableRecordTask()` action.

Note that a `recordTask` is not created by a control point directly; therefore, the input support level below indicates PROHIBITED for all properties.

The table below lists all `recordTask`-related standard defined properties (`recordSchedule`-only properties are omitted from the table – see Appendix B, “AV Working Committee Extended Properties” for the definition of each property) and indicates the support level (PROHIBITED, OPTIONAL, and REQUIRED) in this class for `recordTask` (RT) usage for each property.
Table C-14: `object.recordTask` Base Class Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>RT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base Properties</strong></td>
<td></td>
</tr>
<tr>
<td><code>@id</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>title</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>class</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>additionalStatusInfo</code></td>
<td>O</td>
</tr>
<tr>
<td><code>cdsReference</code></td>
<td>O</td>
</tr>
<tr>
<td><code>cdsReference@link</code></td>
<td>R</td>
</tr>
<tr>
<td><strong>Priority Properties</strong></td>
<td></td>
</tr>
<tr>
<td><code>priority</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>priority@orderedValue</code></td>
<td>O</td>
</tr>
<tr>
<td><code>desiredPriority</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>desiredPriority@type</code></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Output Control Properties</strong></td>
<td></td>
</tr>
<tr>
<td><code>recordDestination</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>recordDestination@mediaType</code></td>
<td>R</td>
</tr>
<tr>
<td><code>recordDestination@targetURL</code></td>
<td>O</td>
</tr>
<tr>
<td><code>recordDestination@preference</code></td>
<td>R</td>
</tr>
<tr>
<td><code>desiredRecordQuality</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>desiredRecordQuality@type</code></td>
<td>✓</td>
</tr>
<tr>
<td><strong>General Properties</strong></td>
<td></td>
</tr>
<tr>
<td><code>recordScheduleID</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>recordedCDSObjectID</code></td>
<td>O</td>
</tr>
<tr>
<td><code>recordedCDSObjectID@link</code></td>
<td>O</td>
</tr>
<tr>
<td><strong>Content ID Related Properties</strong></td>
<td></td>
</tr>
<tr>
<td><code>taskCDSObjectID</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskCDSObjectID@link</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskChannelID</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>taskChannelID@type</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>taskChannelID@distriNetworkName</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskChannelID@distriNetworkID</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskStartDateTime</code></td>
<td>R</td>
</tr>
<tr>
<td><code>taskDuration</code></td>
<td>R</td>
</tr>
<tr>
<td><code>taskProgramCode</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskProgramCode@type</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>recordQuality</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>recordQuality@type</code></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Matched Content Criteria</strong></td>
<td></td>
</tr>
<tr>
<td><code>matchedName</code></td>
<td>O</td>
</tr>
<tr>
<td><code>matchedName@type</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>matchedCDSObjectID</code></td>
<td>O</td>
</tr>
<tr>
<td><code>matchedCDSObjectID@link</code></td>
<td>O</td>
</tr>
<tr>
<td><strong>Matched Qualifying Criteria</strong></td>
<td></td>
</tr>
<tr>
<td><code>matchedRating</code></td>
<td>O</td>
</tr>
<tr>
<td><code>matchedRating@type</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>matchedEpisodeType</code></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Content Control Properties</strong></td>
<td></td>
</tr>
<tr>
<td><code>taskStartDateTimeAdjust</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskDurationAdjust</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskDurationLimit</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskDurationLimit@effect</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>taskChannelMigration</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskTimeMigration</code></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Task State Properties</strong></td>
<td></td>
</tr>
<tr>
<td><code>taskState</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>taskState@phase</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>taskState@startDateTimeMet</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskState@endDateTimeMet</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskState@recording</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskState@someBitsRecorded</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskState@someBitsMissing</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskState@firstBitsRecorded</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskState@lastBitsRecorded</code></td>
<td>O</td>
</tr>
<tr>
<td><code>taskState@fatalError</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>taskState@currentErrors</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>taskState@errorHistory</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>taskState@pendingErrors</code></td>
<td>✓</td>
</tr>
<tr>
<td><code>taskState@infoList</code></td>
<td>✓</td>
</tr>
</tbody>
</table>
Appendix D. EBNF Syntax Definitions (Normative)

The following sections define the syntax used for some of the properties and classes described in the previous sections. The syntax is formally defined using EBNF as described in Section 1.2.3, “Extended Backus-Naur Form”.

D.1 Priority Syntax

Note: Due to possible future extensions, unknown value inputs MUST be gracefully ignored. In this case, the semantics of the “DEFAULT” value MUST be applied.

priority-value ::= standard-value
    | extended-value (* extended-value is only applicable if priority@orderedValue is supported *)

standard-value ::= level | 'DEFAULT'

level ::= ('L' number)

number ::= (* integer (n>0) *)

extended-value ::= 'HIGHEST' | 'LOWEST' | level-hi | level-low | object-id

level-hi ::= level '_HI'

level-low ::= level '_LOW'

object-id ::= (* @id value *)

D.2 Date&time Syntax

sched-start ::= date-time |
    | day-of-yr-time |
    | named-day-time |
    | T-labeled-time |

start-range ::= (date-time | 'NOW') '/' (date-time | 'INFINITY')

date-time-range ::= date-time '/' date-time

duration ::= 'P' [n 'D'] time

duration-long ::= duration | 'INFINITY'

duration-any ::= duration | 'INFINITY' | 'ANY'

duration-adj ::= ('+' | '-') duration

duration-range ::= duration '/' duration-long

date-time ::= yyyy '-' mm '-' dd T-labeled-time

day-of-yr-time ::= mm '-' dd T-labeled-time

named-day-time ::= named-day T-labeled-time

named-day ::= 'MON' | 'TUE' | 'WED' | 'THU' | 'FRI' | 'SAT' | 'SUN' |

  | 'MON-FRI' | 'MON-SAT'

n ::= 1*DIGIT (* non-negative integer *)

yyyy ::= 4DIGIT (* 0001-9999 *)

mm ::= 2DIGIT (* 01-12 *)
\[ \text{dd} \quad ::= \quad \text{2DIGIT} \quad (* \quad 01-28, \quad 01-29, \quad 01-30, \quad 01-31 \quad \\
\text{based on month/year} \quad *) \\
\text{HH} \quad ::= \quad \text{2DIGIT} \quad (* \quad 00-23 \quad *) \\
\text{MM} \quad ::= \quad \text{2DIGIT} \quad (* \quad 00-59 \quad *) \\
\text{SS} \quad ::= \quad \text{2DIGIT} \quad (* \quad 00-59 \quad *) \]

D.3 Class Name Syntax

\[
\text{className} \quad ::= \quad 'OBJECT.' \quad (\text{sName} | \text{tName}) \\
\text{sName} \quad ::= \quad 'RECORDSCHEDULE.' \quad (\text{dName} | \text{qName}) \\
\text{tName} \quad ::= \quad 'RECORDTASK' \quad (\text{'.' shortName})* \\
\text{dName} \quad ::= \quad 'DIRECT.' \quad \text{directName} \quad (\text{'.' shortName})* \\
\text{qName} \quad ::= \quad 'QUERY.' \quad \text{queryName} \quad (\text{'.' shortName})* \\
\text{directName} \quad ::= \quad 'MANUAL'|'CDSEPG'|'CDSNONEPG'|'PROGRAMCODE' \\
\text{queryName} \quad ::= \quad 'CONTENTNAME'|'CONTENTID' \\
\text{shortName} \quad ::= \quad (* \quad \text{valid XML 1.0 name, excluding the characters} \quad \\
\text{'}.' \quad (\text{UTF-8 code 0x2E}) \quad \\
\text{and} \quad \\
\text{'}:' \quad (\text{UTF-8 code 0x3A}) \quad *)
\]
Appendix E. ScheduledRecording Service Relationship to ContentDirectory Service (Informative)

As noted in the specification, the only formal relationship between a ScheduledRecording service and a ContentDirectory service is through the `object.recordSchedule.direct.cdsEPG` and `object.recordSchedule.direct.cdsNonEPG` classes. The reason for keeping the ScheduledRecording service and ContentDirectory service as separate services is because they serve different purposes. The ScheduledRecording service is a service for creating a schedule of recording operations whereas the ContentDirectory service is a service for exposing content and its metadata. Therefore, the only formal dependency on a ContentDirectory service is to accommodate the case where a control point identifies recordable content on a ContentDirectory service and then instructs a sibling ScheduledRecording service to record that content.

Although a ScheduledRecording service and a ContentDirectory service are generally separated at the protocol layer, the two services can often interact in an out-of-band manner to realize some additional usages.

**Showing Recorded Content in a ContentDirectory service:** Vendors who are interested in making recorded content discoverable and network-consumable can expose the recorded content through the associated ContentDirectory service. The exact location where the recorded content will be exposed is determined by the implementation and is vendor-dependent.

**Sending recorded bits to a ContentDirectory service:** One methodology for sending recorded content to a ContentDirectory service (that is completely separate from the ScheduledRecording service) is to do the following: Start the process by having the control point invoke the `ContentDirectory::CreateObject()` action and obtain a `res@importUri` where binary data can be deposited via HTTP-POST. As a second step, the control point uses the `CreateRecordSchedule()` action with the appropriate destination type (`recordDestination = "MyNAS", recordDestination@mediaType = "HDD", recordDestination@targetURL = res@importUri`) to accommodate a URI that accepts HTTP-POST transmissions. When the ScheduledRecording service begins to record (or finishes recording) the ScheduledRecording service implementation can transmit the recorded bits using an HTTP-POST transaction. When the transmission is complete, the ContentDirectory service updates its metadata to allow rendering endpoints to play the content.

**Scheduled recording from an external location:** Vendors who want to use an external location as a source of recordable content can achieve this use case in the following manner. The control point obtains a URI that represents content that can be recorded. The control point creates a manual `recordSchedule` with the appropriate scheduling information and the URI as the input source in the `scheduledChannelID` property. At the instructed time, the ScheduledRecording service will download or stream the content data bytes from the URI to complete the recording.
Appendix F. ScheduledRecording Service Relationship to EPG (Informative)

ScheduledRecording service implementations are NOT REQUIRED to be tied to an Electronic Program Guide (EPG), as demonstrated by the `object.recordSchedule.direct.manual` class. However, the subject of EPG data is an important discussion point for achieving a variety of use cases. This appendix does not exhaustively cover every relationship between a ScheduledRecording service and EPG, but it does discuss how the out-of-band EPG can fit into various use cases.

For scenarios where a control point creates a manual `recordSchedule`, the EPG directly provides information to the user. In some setups, the user may have to read an EPG in order to manually provide the control point with the scheduling and tuning input values. In other setups, the control point may have access to EPG data, allowing the control point to provide a user interface that is focused on the EPG, hiding the control point input values from the user. By design, the `object.recordSchedule.direct.manual` class does not require an EPG on the ScheduledRecording service because the ScheduledRecording service can resolve a manual recording type to discrete `recordTask` instances, without any additional information.

For scenarios where the control point creates a `object.recordSchedule.direct.cdsEPG` or `object.recordSchedule.query.contentName/contentID` class `recordSchedule`, the user still interacts with the EPG in some way. In some setups, the user will need to obtain a well-defined value (program or series ID, program title, etc.) from the EPG. In setups where the control point has access to EPG data, the user may not need to know about those well-defined values. Regardless of how the control point acquires the well-defined values, the ScheduledRecording service still needs to be able to translate this higher-level information into a `recordSchedule` object. In some setups, the ScheduledRecording service will have access to a complete EPG to assist with the creation of individual `recordSchedule` instances. In other setups, the ScheduledRecording service may have access to limited scheduling information on a broadcast stream that allows the ScheduledRecording service to know when to perform a recording operation. Intentionally, these types of recording types are designed around a particular EPG or broadcast system.

For scenarios where the control point creates a `object.recordSchedule.query.contentName` class of `recordSchedule`, the ScheduledRecording service will likely have direct access to EPG data. The reason is that this type of `recordSchedule` allows the control point to specify values that are not well-defined but still convey the desired content for recording. Therefore, the ScheduledRecording service generally needs to have additional intelligence to translate the `recordSchedule` into discrete `recordTask` instances. Often this process will include continually cross-referencing the `recordSchedule` properties with information in an EPG, and generating a `recordTask` instance every time a match is found between the matching criteria of the `recordSchedule` and an EPG item.

For scenarios where the control point creates a `object.recordSchedule.direct.cdsEPG` class of `recordSchedule`, the ScheduledRecording service and its sibling ContentDirectory service will likely have access to some kind of EPG data. The ContentDirectory service uses the EPG to expose recordable content to the control point/user. The user chooses a recordable object and then instructs the ScheduledRecording service to record using the `didl-lite:@id` value of the recordable object. Although it is generally useful for a ContentDirectory service to expose as much scheduling metadata as possible, the EPG data that is exposed by the ContentDirectory service is determined by the ContentDirectory/ScheduledRecording service implementer. As such, the only thing that a user needs is a control point that is capable of representing the recordable objects found in the ContentDirectory service.
Appendix G. AVDT Examples (Informative)

The following sections contain full-fledged examples of AVDT XML Document instances that a particular implementation might return in response to the invocation of the GetAllowedValues() action. All AV Working Committee defined values are printed using the forum character style. All device dependent values are printed using the vendor character style.

Note: These examples may be used as a starting point for real life implementations. Vendors may delete OPTIONAL property definitions that they do not support and add, delete and/or modify device dependent values to match their implementation.

G.1 A_ARG_TYPE_RecordSchedule AVDT Example

Note: This A_ARG_TYPE_RecordSchedule example is marked by a grey background.

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

The following response will be generated:

Response:
GetAllowedValues(
<?xml version="1.0" encoding="UTF-8"?>
<AVDT
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:srs="urn:schemas-upnp-org:av:srs"
 xmlns="urn:schemas-upnp-org:av:avdt"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
 urn:schemas-upnp-org:av:srs http://www.upnp.org/schemas/av/srs.xsd
 urn:schemas-upnp-org:av:avdt http://www.upnp.org/schemas/av/avdt.xsd">
<contextID>
    uuid:device-UUID::urn:schemas-upnp-org:service:ScheduledRecording:1
 </contextID>
<dataStructType>A_ARG_TYPE_RecordSchedule</dataStructType>

<fieldTable>
    <field>
        <name>srs:@id</name>
        <dataType maxSize="256">xsd:string</dataType>
        <minCountTotal>1</minCountTotal>
        <allowedValueDescriptor>
            <allowAny></allowAny>
        </allowedValueDescriptor>
    </field>
    <field>
        <name>srs:title</name>
        <dataType maxSize="128">xsd:string</dataType>
        <minCountTotal>1</minCountTotal>
        <allowedValueDescriptor>
            <allowAny></allowAny>
        </allowedValueDescriptor>
    </field>
</fieldTable>
<dependentField>
   <name>srs:cdsReference</name>
   <anyValue></anyValue>
</dependentField>

<field>
   <name>srs:Priority</name>
   <dataType maxSize="8">xsd:string</dataType>
   <minCountTotal>1</minCountTotal>
   <allowedValueDescriptor>
      <allowedValueList>
         <allowedValue>L1</allowedValue>
         <allowedValue>L2</allowedValue>
         <allowedValue>L3</allowedValue>
      </allowedValueList>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:priority@orderedValue</name>
   <dataType>xsd:unsignedInt</dataType>
   <minCount>1</minCount>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:priority</name>
      </dependentField>
      <minCount>1</minCount>
      <allowedValueRange>
         <minimum>1</minimum>
         <maximum>64</maximum>
         <step>1</step>
      </allowedValueRange>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:desiredPriority</name>
   <dataType maxSize="1024">xsd:string</dataType>
   <minCount>1</minCount>
   <allowedValueDescriptor>
      <dependentField>
         <name>srs:desiredPriority@type</name>
      </dependentField>
      <valueList>
         <value>PREDEF</value>
      </valueList>
   </allowedValueDescriptor>
</field>
<allowedValue>L2_LO</allowedValue>
<allowedValue>L3_HI</allowedValue>
<allowedValue>L3_LO</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
<allowedValueDescriptor>
<dependentField>
  <name>srs:desiredPriority@type</name>
  <valueList>
    <value>OBJECTID</value>
  </valueList>
</dependentField>
</field>

<field>
  <name>srs:desiredPriority@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:desiredPriority</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>PREDEF</allowedValue>
      <allowedValue>OBJECTID</allowedValue>
    </allowedValueList>
    </allowedValueDescriptor>
</field>

<field>
  <name>srs:recordDestination</name>
  <dataType maxSize="1024">xsd:string</dataType>
  <minCountTotal>1</minCountTotal>
  <maxCountTotal>3</maxCountTotal>
  <allowedValueDescriptor>
    <allowedValueList>
      <allowedValue>Hard Disk 1</allowedValue>
      <allowedValue>Hard Disk 2</allowedValue>
      <allowedValue>DVD Drive</allowedValue>
      <allowedValue>Remote Media Jukebox</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:recordDestination@mediaType</name>
  <dataType csv="xsd:string" maxSize="16">xsd:string</dataType>
  <maxCountTotal>3</maxCountTotal>
  <minListSizeTotal>1</minListSizeTotal>
  <maxListSizeTotal>4</maxListSizeTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:recordDestination</name>
      <valueList>
        <value>Hard Disk 1</value>
        <value>Hard Disk 2</value>
      </valueList>
    </dependentField>
  </allowedValueDescriptor>
</field>
<field>
  <name>srs:recordDestination@targetURL</name>
  <dataType>xsd:anyURI</dataType>
  <maxCountTotal>3</maxCountTotal>
  <allowedValueDescriptor>
    <name>srs:recordDestination</name>
    <valueList>
      <value>Remote Media Jukebox</value>
    </valueList>
  </allowedFieldDescriptor>
</field>

<field>
  <name>srs:recordDestination@preference</name>
  <dataType>xsd:unsignedInt</dataType>
</field>
<maxCountTotal>3</maxCountTotal>
<allowedValueDescriptor>
  <dependentField>
    <name>srs:recordDestination</name>
    <anyValue></anyValue>
  </dependentField>
  <minCount>1</minCount>
  <allowedValueRange>
    <minimum>1</minimum>
    <maximum>3</maximum>
    <step>1</step>
  </allowedValueRange>
</allowedValueDescriptor>

<field>
  <name>srs:desiredRecordQuality</name>
  <dataType csv="xsd:string" maxSize="1024">xsd:string</dataType>
  <maxListSizeTotal>UNBOUNDED</maxListSizeTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:desiredRecordQuality@type</name>
      <valueList>
        <value>DEFAULT</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <minListSize>1</minListSize>
    <maxListSize>4</maxListSize>
    <allowedValueList>
      <allowedValue>HD</allowedValue>
      <allowedValue>ED</allowedValue>
      <allowedValue>SD</allowedValue>
      <allowedValue>AUTO</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:desiredRecordQuality@type</name>
  <valueList>
    <value>ATSC</value>
  </valueList>
</field>

<field>
  <name>srs:desiredRecordQuality@type</name>
  <valueList>
    <value>1080p30</value>
    <value>1080p24</value>
    <value>1080i60</value>
    <value>720p60</value>
    <value>720p30</value>
    <value>720p24</value>
    <value>480p60</value>
    <value>480p30</value>
    <value>480p24</value>
    <value>480i60</value>
    <value>AUTO</value>
  </valueList>
</field>
<field>
  <name>srs:desiredRecordQuality@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:desiredRecordQuality</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>QLEVEL</allowedValue>
      <allowedValue>Q1</allowedValue>
      <allowedValue>Q2</allowedValue>
      <allowedValue>Q3</allowedValue>
      <allowedValue>AUTO</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:desiredRecordQuality@type</name>
  <dataType maxSize="16" xsd:string></dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:desiredRecordQuality</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>DEFAULT</allowedValue>
      <allowedValue>ATSC</allowedValue>
      <allowedValue>QLEVEL</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledCDSObjectID</name>
  <dataType maxSize="1024">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledCDSObjectID@link</name>
  <dataType maxSize="256">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:scheduledCDSObjectID</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>
<field>
  <name>srs:scheduledChannelID</name>
  <dataType maxSize="256">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledChannelID@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:scheduledChannelID</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>ANALOG</allowedValue>
      <allowedValue>DIGITAL</allowedValue>
      <allowedValue>FREQUENCY</allowedValue>
      <allowedValue>SI</allowedValue>
      <allowedValue>LINE</allowedValue>
      <allowedValue>NETWORK</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledChannelID@distriNetworkName</name>
  <dataType maxSize="32">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:scheduledChannelID</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedAny></allowedAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledChannelID@distriNetworkID</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:scheduledChannelID</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedAny></allowedAny>
  </allowedValueDescriptor>
</field>
<allowAny></allowAny>
</allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledStartDateTime</name>
  <dataType maxSize="64">xsd:string</dataType>
  <minCountTotal>1</minCountTotal>
  <maxCountTotal>2</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
      </valueList>
      <allowAny></allowAny>
    </dependentField>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledDuration</name>
  <dataType maxSize="64">xsd:string</dataType>
  <minCount>1</minCount>
  <allowAny></allowAny>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
      </valueList>
    </dependentField>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledProgramCode</name>
  <dataType maxSize="64">xsd:string</dataType>
  <minCount>1</minCount>
  <allowAny></allowAny>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE</value>
      </valueList>
    </dependentField>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledProgramCode@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <minCount>1</minCount>
  <allowAny></allowAny>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:scheduledProgramCode</name>
    </dependentField>
  </allowedValueDescriptor>
</field>
<field>
  <name>srs:matchingID@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchingID</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>SI_PROGRAMID</allowedValue>
      <allowedValue>SI_SERIESID</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingChannelID</name>
  <dataType maxSize="256">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
      </valueList>
    </dependentField>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingChannelID@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchingChannelID</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>ANALOG</allowedValue>
      <allowedValue>DIGITAL</allowedValue>
      <allowedValue>FREQUENCY</allowedValue>
      <allowedValue>SI</allowedValue>
      <allowedValue>LINE</allowedValue>
      <allowedValue>NETWORK</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingChannelID@distriNetworkName</name>
  <dataType maxSize="32">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchingChannelID</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>ANALOG</allowedValue>
      <allowedValue>DIGITAL</allowedValue>
      <allowedValue>FREQUENCY</allowedValue>
      <allowedValue>SI</allowedValue>
      <allowedValue>LINE</allowedValue>
      <allowedValue>NETWORK</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>
<allowedValueList>
  <allowedValue>RP</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
<dependentField>
  <name>srs:class</name>
  <valueList>
    <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
    <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
  </valueList>
</dependentField>
<dependentField>
  <name>srs:matchingRatingLimit@type</name>
  <valueList>
    <value>TVGUIDELINES.ORG</value>
  </valueList>
</dependentField>
<minCount>1</minCount>
<allowedValueList>
  <allowedValue>TV-Y</allowedValue>
  <allowedValue>TV-Y7</allowedValue>
  <allowedValue>TV-Y7FV</allowedValue>
  <allowedValue>TV-G</allowedValue>
  <allowedValue>TV-PG</allowedValue>
  <allowedValue>TV-14</allowedValue>
  <allowedValue>TV-MA</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingRatingLimit@type</name>
  <dataType maxSize="32">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchingRatingLimit</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>MPAA.ORG</allowedValue>
      <allowedValue>RIAA.ORG</allowedValue>
      <allowedValue>ESRB.ORG</allowedValue>
      <allowedValue>TVGUIDELINES.ORG</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingEpisodeType</name>
  <dataType maxSize="9">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
    </allowedValueList>
  </allowedValueDescriptor>
</field>
<field>
  <name>srs:durationLimitEffect</name>
  <dataType maxSize="8">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:durationLimit</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>FIRST</allowedValue>
      <allowedValue>LAST</allowedValue>
      <allowedValue>SKIP</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:channelMigration</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
      </valueList>
    </dependentField>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:timeMigration</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
      </valueList>
    </dependentField>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:allowDuplicates</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
  </allowedValueDescriptor>
</field>
<valueList>
  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
  <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
</valueList>
</dependentField>
<allowAny></allowAny>
</allowedValueDescriptor>
</field>

<field>
  <name>srs:persistedRecordings</name>
  <dataType>xsd:unsignedInt</dataType>
  <allowedValueDescriptor>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:persistedRecordings@latest</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:persistedRecordings</name>
      <anyValue></anyValue>
    </dependentField>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:persistedRecordings@preAllocation</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:persistedRecordings</name>
      <anyValue></anyValue>
    </dependentField>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:persistedRecordings@storedLifetime</name>
  <dataType maxsize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:persistedRecordings</name>
      <anyValue></anyValue>
    </dependentField>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduleState</name>
  <dataType maxsize="16">xsd:string</dataType>
  <minCountTotal>1</minCountTotal>
  <allowedValueDescriptor>
    <allowedValueList>
      <allowedValue>OPERATIONAL</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>
<allowedValueList>
  <allowedValue>
    COMPLETED
  </allowedValue>
  <allowedValue>
    ERROR
  </allowedValue>
</allowedValueList>

<field>
  <name>srs:scheduleState@currentErrors</name>
  <dataType csv="xsd:int">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:scheduleState</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <maxListSize>UNBOUNDED</maxListSize>
    <allowedValueRange>
      <minimum>100</minimum>
      <maximum>105</maximum>
      <step>1</step>
    </allowedValueRange>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduleState@abnormalTasksExist</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:scheduleState</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

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

<field>
  <name>srs:totalCreatedRecordTasks</name>
  <dataType>xsd:unsignedInt</dataType>
  <allowedValueDescriptor>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:totalCompletedRecordTasks</name>
  <dataType>xsd:unsignedInt</dataType>
  <allowedValueDescriptor>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>
G.2 **A ARG TYPE_RecordTask** AVDT Example

Note: This *A ARG TYPE_RecordTask* example is marked by a light turquoise background.

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

The following response will be generated:

**Response:**
GetAllowedValues(
<?xml version="1.0" encoding="UTF-8"?>
<AVDT
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:srs="urn:schemas-upnp-org:av:srs"
xmlns="urn:schemas-upnp-org:av:avdt"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
  urn:schemas-upnp-org:av:srs
  http://www.upnp.org/schemas/av/srs.xsd
  urn:schemas-upnp-org:av:avdt
  http://www.upnp.org/schemas/av/avdt.xsd">
<contextID>
  uuid:device-UUID:urn:schemas-upnp-org:service:ScheduledRecording:1
</contextID>
<dataStructType>A ARG TYPE_RecordTask</dataStructType>

<fieldTable>
  <field>
    <name>srs:@id</name>
    <dataType maxSize="256">xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
      <allowAny/>
    </allowedValueDescriptor>
  </field>

  <field>
    <name>srs:title</name>
    <dataType maxSize="128">xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
      <allowAny/>
    </allowedValueDescriptor>
  </field>

  <field>
    <name>srs:class</name>
    <dataType maxSize="64">xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
      <allowedValueList>
        <allowedValue>OBJECT.RECORDTASK</allowedValue>
      </allowedValueList>
    </allowedValueDescriptor>
  </field>
</fieldTable>
<field>
  <name>srs:additionalInfo</name>
  <dataType maxSize="1024">xsd:string</dataType>
  <allowedValueDescriptor>
    <allowAny/></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:cdsReference</name>
  <dataType maxSize="8192">xsd:string</dataType>
  <maxCountTotal>2</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskCDSObjectID</name>
      <anyValue/></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny/></allowAny>
  </allowedValueDescriptor>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:recordedCDSObjectID</name>
      <anyValue/></anyValue>
    </dependentField>
    <allowAny/></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:cdsReference@link</name>
  <dataType maxSize="1024">xsd:string</dataType>
  <maxCountTotal>2</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:cdsReference</name>
      <anyValue/></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny/></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:Priority</name>
  <dataType maxSize="8">xsd:string</dataType>
  <minCountTotal>1</minCountTotal>
  <allowedValueDescriptor>
    <allowedValueList>
      <allowedValue>L1</allowedValue>
      <allowedValue>L2</allowedValue>
      <allowedValue>L3</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>
<field>
    <name>srs:priority@orderedValue</name>
    <dataType>xsd:unsignedInt</dataType>
    <allowedValueDescriptor>
        <allowedValueRange>
            <minimum>1</minimum>
            <maximum>64</maximum>
            <step>1</step>
        </allowedValueRange>
        </allowedValueDescriptor>
    </field>

    <field>
        <name>srs:desiredPriority</name>
        <dataType maxSize="1024">xsd:string</dataType>
        <allowedValueDescriptor>
            <dependentField>
                <name>srs:desiredPriority@type</name>
                <valueList>
                    <value>PREDEF</value>
                </valueList>
            </dependentField>
            <allowedValueList>
                <allowedValue>DEFAULT</allowedValue>
                <allowedValue>L1</allowedValue>
                <allowedValue>L2</allowedValue>
                <allowedValue>L3</allowedValue>
                <allowedValue>HIGHEST</allowedValue>
                <allowedValue>LOWEST</allowedValue>
                <allowedValue>L1_HI</allowedValue>
                <allowedValue>L1_LO</allowedValue>
                <allowedValue>L2_HI</allowedValue>
                <allowedValue>L2_LO</allowedValue>
                <allowedValue>L3_HI</allowedValue>
                <allowedValue>L3_LO</allowedValue>
            </allowedValueList>
        </allowedValueDescriptor>
        <allowedValueDescriptor>
            <dependentField>
                <name>srs:desiredPriority@type</name>
                <valueList>
                    <value>OBJECTID</value>
                </valueList>
            </dependentField>
            <allowAny></allowAny>
        </allowedValueDescriptor>
        </field>

        <field>
            <name>srs:desiredPriority@type</name>
            <dataType maxSize="16">xsd:string</dataType>
            <allowedValueDescriptor>
                <dependentField>
                    <name>srs:desiredPriority</name>
                    <anyValue></anyValue>
                </dependentField>
                <minCount>1</minCount>
                <allowedValueList>
                    <allowedValue>PREDEF</allowedValue>
                    <allowedValue>OBJECTID</allowedValue>
                </allowedValueList>
            </allowedValueDescriptor>
        </field>
<field>
  <name>srs:recordDestination</name>
  <dataType maxSize="1024">xsd:string</dataType>
  <minCountTotal>1</minCountTotal>
  <maxCountTotal>3</maxCountTotal>
  <allowedValueDescriptor>
    <allowedValue>Hard Disk 1</allowedValue>
    <allowedValue>Hard Disk 2</allowedValue>
    <allowedValue>DVD Drive</allowedValue>
    <allowedValue>Remote Media Jukebox</allowedValue>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:recordDestination@mediaType</name>
  <dataType csv="xsd:string" maxSize="16">xsd:string</dataType>
  <maxCountTotal>3</maxCountTotal>
  <minListSizeTotal>1</minListSizeTotal>
  <maxListSizeTotal>4</maxListSizeTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:recordDestination</name>
      <valueList>
        <value>HardDisk 1</value>
        <value>HardDisk 2</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <minListSize>1</minListSize>
    <allowedValueList>
      <allowedValue>HDD</allowedValue>
      <allowedValue>DVD+RW</allowedValue>
      <allowedValue>DVD-RW</allowedValue>
      <allowedValue>DVD-R</allowedValue>
      <allowedValue>DVD+R</allowedValue>
      <allowedValue>CD-R</allowedValue>
      <allowedValue>CD-RW</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>
<valueList>
  <value>Remote Media Jukebox</value>
</valueList>
</dependentField>

<minCount>1</minCount>
<minListSize>1</minListSize>
<maxListSize>2</maxListSize>
<allowedValueList>
  <allowedValue>CD-R</allowedValue>
  <allowedValue>CD-RW</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
</field>

<field>
  <name>srs:recordDestination@targetURL</name>
  <dataType>xsd:anyURI</dataType>
  <maxCountTotal>3</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:recordDestination</name>
      <anyValue></anyValue>
    </dependentField>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:recordDestination@preference</name>
  <dataType>xsd:unsignedInt</dataType>
  <maxCountTotal>3</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:recordDestination</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueRange>
      <minimum>1</minimum>
      <maximum>3</maximum>
      <step>1</step>
    </allowedValueRange>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:desiredRecordQuality</name>
  <dataType csv="xsd:string" maxSize="1024">xsd:string</dataType>
  <maxListSizeTotal>UNBOUNDED</maxListSizeTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:desiredRecordQuality@type</name>
      <valueList>
        <value>DEFAULT</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <minListSize>1</minListSize>
    <maxListSize>4</maxListSize>
    <allowedValueList>
<allowedValue>HD</allowedValue>
<allowedValue>ED</allowedValue>
<allowedValue>SD</allowedValue>
<allowedValue>AUTO</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
<allowedValueDescriptor>
<dependentField>
  <name>srs:desiredRecordQuality@type</name>
  <valueList>
    <value>ATSC</value>
  </valueList>
</dependentField>

<minCount>1</minCount>
<minListSize>1</minListSize>
<maxListSize>11</maxListSize>
<allowedValueList>
  <allowedValue>1080p30</allowedValue>
  <allowedValue>1080p24</allowedValue>
  <allowedValue>1080i60</allowedValue>
  <allowedValue>720p60</allowedValue>
  <allowedValue>720p30</allowedValue>
  <allowedValue>720p24</allowedValue>
  <allowedValue>480p60</allowedValue>
  <allowedValue>480p30</allowedValue>
  <allowedValue>480p24</allowedValue>
  <allowedValue>480i60</allowedValue>
  <allowedValue>AUTO</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
<allowedValueDescriptor>
<dependentField>
  <name>srs:desiredRecordQuality@type</name>
  <valueList>
    <value>QLEVEL</value>
  </valueList>
</dependentField>

<minCount>1</minCount>
<minListSize>1</minListSize>
<maxListSize>4</maxListSize>
<allowedValueList>
  <allowedValue>Q1</allowedValue>
  <allowedValue>Q2</allowedValue>
  <allowedValue>Q3</allowedValue>
  <allowedValue>AUTO</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
</field>

<field>
  <name>srs:desiredRecordQuality@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:desiredRecordQuality</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
    </allowedValueList>
  </allowedValueDescriptor>
</field>
<field>
    <name>srs:recordScheduleID</name>
    <dataType maxSize="256">xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
        <allowAny/>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:recordedCDSObjectID</name>
    <dataType maxSize="8192">xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
        <allowAny/>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:recordedCDSObjectID@link</name>
    <dataType maxSize="1024">xsd:string</dataType>
    <allowedValueDescriptor>
        <dependentField>
            <name>srs:recordedCDSObjectID</name>
            <anyValue/>
        </dependentField>
        <allowAny/>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:taskCDSObjectID</name>
    <dataType maxSize="8192">xsd:string</dataType>
    <allowedValueDescriptor>
        <allowAny/>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:taskCDSObjectID@link</name>
    <dataType maxSize="1024">xsd:string</dataType>
    <allowedValueDescriptor>
        <dependentField>
            <name>srs:taskCDSObjectID</name>
            <anyValue/>
        </dependentField>
        <allowAny/>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:taskChannelID</name>
    <dataType maxSize="256">xsd:string</dataType>
<field>
    <name>srs:taskChannelID@type</name>
    <dataType maxSize="16">xsd:string</dataType>
    <allowedValueDescriptor>
        <dependentField>
            <name>srs:taskChannelID</name>
            <anyValue></anyValue>
        </dependentField>
        <minCount>1</minCount>
        <allowedValueList>
            <allowedValue>ANALOG</allowedValue>
            <allowedValue>DIGITAL</allowedValue>
            <allowedValue>FREQUENCY</allowedValue>
            <allowedValue>SI</allowedValue>
            <allowedValue>LINE</allowedValue>
            <allowedValue>NETWORK</allowedValue>
        </allowedValueList>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:taskChannelID@distriNetworkName</name>
    <dataType maxSize="32">xsd:string</dataType>
    <allowedValueDescriptor>
        <dependentField>
            <name>srs:taskChannelID</name>
            <anyValue></anyValue>
        </dependentField>
        <allowAny></allowAny>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:taskChannelID@distriNetworkID</name>
    <dataType maxSize="16">xsd:string</dataType>
    <allowedValueDescriptor>
        <dependentField>
            <name>srs:taskChannelID</name>
            <anyValue></anyValue>
        </dependentField>
        <allowAny></allowAny>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:taskStartDateTime</name>
    <dataType maxSize="64">xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
        <allowAny></allowAny>
    </allowedValueDescriptor>
</field>
<field>
  <name>srs:taskDuration</name>
  <dataType maxSize="64">xsd:string</dataType>
  <minCountTotal>1</minCountTotal>
  <allowedValueDescriptor>
    <allowAny/>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskProgramCode</name>
  <dataType maxSize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <allowAny/>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskProgramCode@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskProgramCode</name>
      <anyValue/>
    </dependentField>
    <minCount>1</minCount>
    <allowAny/>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:recordQuality</name>
  <dataType maxSize="16">xsd:string</dataType>
  <minCountTotal>3</minCountTotal>
  <maxCountTotal>3</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:recordQuality@type</name>
      <valueList>
        <value>DEFAULT</value>
      </valueList>
    </dependentField>
    <dependentField>
      <name>srs:taskState@phase</name>
      <valueList>
        <value>IDLE</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>HD</allowedValue>
      <allowedValue>ED</allowedValue>
      <allowedValue>SD</allowedValue>
      <allowedValue>UNKNOWN</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:recordQuality@type</name>
  <valueList>
    <value>DEFAULT</value>
  </valueList>
</field>
<dependentField>
  <name>srs:taskState@phase</name>
  <valueList>
    <value>ACTIVE</value>
    <value>DONE</value>
  </valueList>
</dependentField>

<minCount>1</minCount>
<allowedValueList>
  <allowedValue>HD</allowedValue>
  <allowedValue>ED</allowedValue>
  <allowedValue>SD</allowedValue>
</allowedValueList>
</allowedValueDescriptor>

<allowedValueDescriptor>
  <dependentField>
    <name>srs:recordQuality@type</name>
    <valueList>
      <value>ATSC</value>
    </valueList>
  </dependentField>

  <dependentField>
    <name>srs:taskState@phase</name>
    <valueList>
      <value>IDLE</value>
    </valueList>
  </dependentField>

  <minCount>1</minCount>
  <allowedValueList>
    <allowedValue>1080p30</allowedValue>
    <allowedValue>1080p24</allowedValue>
    <allowedValue>1080i60</allowedValue>
    <allowedValue>1080i24</allowedValue>
    <allowedValue>720p60</allowedValue>
    <allowedValue>720p30</allowedValue>
    <allowedValue>720p24</allowedValue>
    <allowedValue>720i60</allowedValue>
    <allowedValue>720i30</allowedValue>
    <allowedValue>480p60</allowedValue>
    <allowedValue>480p30</allowedValue>
    <allowedValue>480p24</allowedValue>
    <allowedValue>480i60</allowedValue>
    <allowedValue>480i30</allowedValue>
    <allowedValue>UNKNOWN</allowedValue>
  </allowedValueList>
</allowedValueDescriptor>

<allowedValueDescriptor>
  <dependentField>
    <name>srs:recordQuality@type</name>
    <valueList>
      <value>ATSC</value>
    </valueList>
  </dependentField>

  <dependentField>
    <name>srs:taskState@phase</name>
    <valueList>
      <value>ACTIVE</value>
      <value>DONE</value>
    </valueList>
  </dependentField>

  <minCount>1</minCount>
  <allowedValueList>
    <allowedValue>480p60</allowedValue>
    <allowedValue>480p30</allowedValue>
    <allowedValue>480p24</allowedValue>
    <allowedValue>480i60</allowedValue>
    <allowedValue>480i30</allowedValue>
    <allowedValue>UNKNOWN</allowedValue>
  </allowedValueList>
</allowedValueDescriptor>
<allowedValueList>
  <allowedValue>1080p30</allowedValue>
  <allowedValue>1080p24</allowedValue>
  <allowedValue>1080i60</allowedValue>
  <allowedValue>720p60</allowedValue>
  <allowedValue>720p30</allowedValue>
  <allowedValue>480p60</allowedValue>
  <allowedValue>480p30</allowedValue>
  <allowedValue>480i60</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
<allowedValueDescriptor>
  <dependentField>
    <name>srs:recordQuality@type</name>
    <valueList>
      <value>QLEVEL</value>
    </valueList>
  </dependentField>
  <dependentField>
    <name>srs:taskState@phase</name>
    <valueList>
      <value>IDLE</value>
    </valueList>
  </dependentField>
  <minCount>1</minCount>
  <allowedValueList>
    <allowedValue>Q1</allowedValue>
    <allowedValue>Q2</allowedValue>
    <allowedValue>Q3</allowedValue>
    <allowedValue>UNKNOWN</allowedValue>
  </allowedValueList>
</allowedValueDescriptor>
<allowedValueDescriptor>
  <dependentField>
    <name>srs:recordQuality@type</name>
    <valueList>
      <value>QLEVEL</value>
    </valueList>
  </dependentField>
  <dependentField>
    <name>srs:taskState@phase</name>
    <valueList>
      <value>ACTIVE</value>
      <value>DONE</value>
    </valueList>
  </dependentField>
  <minCount>1</minCount>
  <allowedValueList>
    <allowedValue>Q1</allowedValue>
    <allowedValue>Q2</allowedValue>
    <allowedValue>Q3</allowedValue>
  </allowedValueList>
</allowedValueDescriptor>
</field>

<field>
  <name>srs:recordQuality@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
<dependentField>
  <name>srs:recordQuality</name>
  <anyValue></anyValue>
</dependentField>

<minCount>3</minCount>
<maxCount>3</maxCount>
<allowedValueList>
  <allowedValue>DEFAULT</allowedValue>
  <allowedValue>ATSC</allowedValue>
  <allowedValue>QLEVEL</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
</field>

<field>
  <name>srs:matchedName</name>
  <dataType maxSize="128">xsd:string</dataType>
  <allowedValueDescriptor>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchedName@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchedName</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>PROGRAM</allowedValue>
      <allowedValue>SERIES</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchedID</name>
  <dataType maxSize="256">xsd:string</dataType>
  <allowedValueDescriptor>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchedID@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchedID</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>SI_PROGRAMID</allowedValue>
      <allowedValue>SI_SERIESID</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>
<field>
  <name>srs:matchedRating</name>
  <dataType maxSize="16">xsd:string</dataType>
  <maxCountTotal>2</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchedRating@type</name>
      <valueList>
        <value>MPAA.ORG</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>G</allowedValue>
      <allowedValue>PG</allowedValue>
      <allowedValue>PG-13</allowedValue>
      <allowedValue>R</allowedValue>
      <allowedValue>NC-17</allowedValue>
      <allowedValue>NR</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchedRating@type</name>
      <valueList>
        <value>RIAA.ORG</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>PA-EC</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchedRating@type</name>
      <valueList>
        <value>ESRB.ORG</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>EC</allowedValue>
      <allowedValue>E</allowedValue>
      <allowedValue>E10+</allowedValue>
      <allowedValue>T</allowedValue>
      <allowedValue>M</allowedValue>
      <allowedValue>AO</allowedValue>
      <allowedValue>RP</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchedRating@type</name>
      <valueList>
        <value>TVGUIDELINES.ORG</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>TV-PG</allowe
<field>
  <name>srs:matchedRating@type</name>
  <dataType maxSize="32">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchedRating</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>TV-Y</allowedValue>
      <allowedValue>TV-Y7</allowedValue>
      <allowedValue>TV-Y7FV</allowedValue>
      <allowedValue>TV-G</allowedValue>
      <allowedValue>TV-PG</allowedValue>
      <allowedValue>TV-14</allowedValue>
      <allowedValue>TV-MA</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchedEpisodeType</name>
  <dataType maxSize="8">xsd:string</dataType>
  <allowedValueDescriptor>
    <allowedValueList>
      <allowedValue>ALL</allowedValue>
      <allowedValue-FIRST_RUN</allowedValue>
      <allowedValue-REPEAT</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskStartDateTimeAdjust</name>
  <dataType maxSize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskDurationAdjust</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>
<field>
  <name>srs:taskDurationLimit</name>
  <dataType maxSize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <allowAny/>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskDurationLimit@effect</name>
  <dataType maxSize="8">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskDurationLimit</name>
      <anyValue/>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>FIRST</allowedValue>
      <allowedValue>LAST</allowedValue>
      <allowedValue>Skip</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskChannelMigration</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <allowAny/>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskTimeMigration</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <allowAny/>
  </allowedValueDescriptor>
</field>

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

<field>
  <name>srs:taskState@phase</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskState</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>
        IDLE
      </allowedValue>
      <allowedValue>
        ACTIVE
      </allowedValue>
      <allowedValue>
        DONE
      </allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskState@startDateTimeMet</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskState</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskState@endDateTimeMet</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskState</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskState@recording</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskState</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>
<field>
  <name>srs:taskState@someBitRecorded</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskState</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskState@someBitsMissing</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskState</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskState@firstBitsRecorded</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskState</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskState@lastBitsRecorded</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskState</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
<name>srs:taskState@fatalError</name>
<dataType>xsd:boolean</dataType>
<allowedValueDescriptor>
    <dependentField>
        <name>srs:taskState</name>
        <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowAny></allowAny>
</allowedValueDescriptor>
</field>

<field>
    <name>srs:taskState@currentErrors</name>
    <dataType csv="xsd:int" maxSize="64">xsd:string</dataType>
    <allowedValueDescriptor>
        <dependentField>
            <name>srs:taskState</name>
            <anyValue></anyValue>
        </dependentField>
        <minCount>1</minCount>
        <allowedValueList>
            <allowedValue></allowedValue>
            <allowedValue>100</allowedValue>
            <allowedValue>101</allowedValue>
            <allowedValue>102</allowedValue>
        </allowedValueList>
        <allowedValueRange>
            <minimum>200</minimum>
            <maximum>204</maximum>
            <step>1</step>
        </allowedValueRange>
        <allowedValueRange>
            <minimum>300</minimum>
            <maximum>307</maximum>
            <step>1</step>
        </allowedValueRange>
        <allowedValueRange>
            <minimum>400</minimum>
            <maximum>404</maximum>
            <step>1</step>
        </allowedValueRange>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:taskState@errorHistory</name>
    <dataType csv="xsd:int" maxSize="64">xsd:string</dataType>
    <allowedValueDescriptor>
        <dependentField>
            <name>srs:taskState</name>
            <anyValue></anyValue>
        </dependentField>
        <minCount>1</minCount>
        <allowedValueList>
            <allowedValue></allowedValue>
            <allowedValue>100</allowedValue>
            <allowedValue>101</allowedValue>
            <allowedValue>102</allowedValue>
        </allowedValueList>
        <allowedValueRange>
            <minimum>200</minimum>
            <maximum>204</maximum>
            <step>1</step>
        </allowedValueRange>
        <allowedValueRange>
            <minimum>300</minimum>
            <maximum>307</maximum>
            <step>1</step>
        </allowedValueRange>
        <allowedValueRange>
            <minimum>400</minimum>
            <maximum>404</maximum>
            <step>1</step>
        </allowedValueRange>
    </allowedValueDescriptor>
</field>
Additional vendor defined values go here -->
</allowedValueDescriptor>
</field>

<field>
  <name>srs:taskState@pendingErrors</name>
  <dataType csv="xsd:int" maxSize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskState</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue/>
      <allowedValue>100</allowedValue>
      <allowedValue>101</allowedValue>
      <allowedValue>102</allowedValue>
      <!-- Additional vendor defined values go here -->
    </allowedValueList>
    <allowedValueRange>
      <minimum>200</minimum>
      <maximum>204</maximum>
      <step>1</step>
    </allowedValueRange>
    <allowedValueRange>
      <minimum>300</minimum>
      <maximum>307</maximum>
      <step>1</step>
    </allowedValueRange>
    <allowedValueRange>
      <minimum>400</minimum>
      <maximum>404</maximum>
      <step>1</step>
    </allowedValueRange>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:taskState@infoList</name>
  <dataType csv="xsd:int" maxSize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:taskState</name>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue/>
      <allowedValue>100</allowedValue>
      <allowedValue>101</allowedValue>
      <allowedValue>102</allowedValue>
      <!-- Additional vendor defined values go here -->
    </allowedValueList>
    <allowedValueRange>
      <minimum>200</minimum>
      <maximum>204</maximum>
      <step>1</step>
    </allowedValueRange>
    <allowedValueRange>
      <minimum>300</minimum>
      <maximum>307</maximum>
      <step>1</step>
    </allowedValueRange>
    <allowedValueRange>
      <minimum>400</minimum>
      <maximum>404</maximum>
      <step>1</step>
    </allowedValueRange>
  </allowedValueDescriptor>
</field>
G.3 **A_ARG_TYPE_RecordScheduleParts** AVDT Example

Note: This **A_ARG_TYPE_RecordScheduleParts** example is marked by a white background.

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

The following response will be generated:

**Response:**
GetAllowedValues(
<?xml version="1.0" encoding="UTF-8"?>
<AVDT>
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:srs="urn:schemas-upnp-org:av:srs"
xmlns="urn:schemas-upnp-org:av:avdt"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=""urn:schemas-upnp-org:av:srs"
http://www.upnp.org/schemas/av/srs.xsd
urn:schemas-upnp-org:av:avdt"
http://www.upnp.org/schemas/av/avdt.xsd">
<contextID>
	uuid:device-UUID::urn:schemas-upnp-org:service:ScheduledRecording:1
</contextID>
<dataStructType>A_ARG_TYPE_RecordScheduleParts</dataStructType>
</fieldTable>
</AVDT>
<field>
    <name>srs:oid</name>
    <dataType maxSize="256">xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
        <allowedValueList>
            <allowedValue></allowedValue>
        </allowedValueList>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:title</name>
    <dataType maxSize="128">xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
        <allowAny></allowAny>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:class</name>
    <dataType maxSize="64">xsd:string</dataType>
    <minCountTotal>1</minCountTotal>
    <allowedValueDescriptor>
        <allowedValueList>
            <allowedValue>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</allowedValue>
            <allowedValue>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</allowedValue>
            <allowedValue>OBJECT.RECORDSCHEDULE.DIRECT.CDSONEPG</allowedValue>
            <allowedValue>OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE</allowedValue>
            <allowedValue>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</allowedValue>
            <allowedValue>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</allowedValue>
        </allowedValueList>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:desiredPriority</name>
    <dataType maxSize="1024">xsd:string</dataType>
    <dependentField defaultDependency="1">
        <name>srs:desiredPriority@type</name>
        <valueList>
            <value>PREDEF</value>
        </valueList>
    </dependentField>
    <minCount>1</minCount>
    <defaultValue>DEFAULT</defaultValue>
</field>
<allowedValueList>
  <allowedValue>DEFAULT</allowedValue>
  <allowedValue>L1</allowedValue>
  <allowedValue>L2</allowedValue>
  <allowedValue>L3</allowedValue>
  <allowedValue>HIGHEST</allowedValue>
  <allowedValue>LOWEST</allowedValue>
  <allowedValue>L1_HI</allowedValue>
  <allowedValue>L1_LO</allowedValue>
  <allowedValue>L2_HI</allowedValue>
  <allowedValue>L2_LO</allowedValue>
  <allowedValue>L3_HI</allowedValue>
  <allowedValue>L3_LO</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
<allowedValueDescriptor>
  <dependentField>
    <name>srs:desiredPriority@type</name>
    <valueList>
      <value>OBJECTID</value>
    </valueList>
  </dependentField>
  <minCount>1</minCount>
  <allowAny/></allowAny>
</allowedValueDescriptor>
</field>

<field>
  <name>srs:desiredPriority@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField defaultDependency="1">
      <name>srs:desiredPriority</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <defaultValue>PREDIF</defaultValue>
    <allowedValueList>
      <allowedValue>PREDIF</allowedValue>
      <allowedValue>OBJECTID</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:recordDestination</name>
  <dataType maxSize="1024">xsd:string</dataType>
  <maxCountTotal>3</maxCountTotal>
  <allowedValueDescriptor>
    <defaultValue>Hard Disk 2</defaultValue>
    <allowedValueList>
      <allowedValue>Hard Disk 1</allowedValue>
      <allowedValue>Hard Disk 2</allowedValue>
      <allowedValue>DVD Drive</allowedValue>
      <allowedValue>Remote Media Jukebox</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>
<field>
  <name>srs:recordDestination@mediaType</name>
  <dataType csv="xsd:string" maxSize="16">xsd:string</dataType>
  <maxCountTotal>3</maxCountTotal>
  <maxListSizeTotal>4</maxListSizeTotal>
  <allowedValueDescriptor>
    <dependentField defaultDependency="1">
      <name>srs:recordDestination</name>
      <valueList>
        <value>HardDisk 1</value>
        <value>HardDisk 2</value>
      </valueList>
    </dependentField>
    <maxListSize>1</maxListSize>
    <defaultValue>HDD</defaultValue>
    <allowedValueList>
      <allowedValue>HDD</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:recordDestination</name>
      <valueList>
        <value>DVD Drive</value>
      </valueList>
    </dependentField>
    <maxListSize>4</maxListSize>
    <allowedValueList>
      <allowedValue>DVD+RW</allowedValue>
      <allowedValue>DVD-RW</allowedValue>
      <allowedValue>DVD-R</allowedValue>
      <allowedValue>DVD+R</allowedValue>
      <allowedValue>CD-R</allowedValue>
      <allowedValue>CD-RW</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:recordDestination</name>
      <valueList>
        <value>Network Jukebox Recorder</value>
      </valueList>
    </dependentField>
    <maxListSize>2</maxListSize>
    <defaultValue>CD-R</defaultValue>
    <allowedValueList>
      <allowedValue>CD-R</allowedValue>
      <allowedValue>CD-RW</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:recordDestination@targetURL</name>
  <dataType>xsd:anyURI</dataType>
  <maxCountTotal>3</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:recordDestination</name>
      <anyValue></anyValue>
    </dependentField>
  </allowedValueDescriptor>
</field>
<field>
   <name>srs:recordDestination@preference</name>
   <dataType>xsd:unsignedInt</dataType>
   <maxCountTotal>3</maxCountTotal>
   <allowedValueDescriptor>
      <dependentField defaultDependency="I">srs:recordDestination</dependentField>
      <defaultValue>2</defaultValue>
      <allowedValueRange>
         <minimum>1</minimum>
         <maximum>3</maximum>
         <step>1</step>
      </allowedValueRange>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:desiredRecordQuality</name>
   <dataType csv="xsd:string" maxSize="1024">xsd:string</dataType>
   <maxListSizeTotal>UNBOUNDED</maxListSizeTotal>
   <allowedValueDescriptor>
      <dependentField defaultDependency="I">
         <name>srs:desiredRecordQuality@type</name>
         <valueList>
            <value>DEFAULT</value>
            <value>HD</value>
            <value>ED</value>
            <value>SD</value>
            <value>AUTO</value>
         </valueList>
      </dependentField>
      <minCount>1</minCount>
      <minListSize>1</minListSize>
      <maxListSize>4</maxListSize>
      <defaultValue>AUTO</defaultValue>
   </allowedValueDescriptor>
</field>

<field>
   <name>srs:desiredRecordQuality@type</name>
   <dataType csv="xsd:string" maxSize="1024">xsd:string</dataType>
   <maxListSizeTotal>UNBOUNDED</maxListSizeTotal>
   <allowedValueDescriptor>
      <dependentField defaultDependency="I">
         <name>srs:desiredRecordQuality@type</name>
         <valueList>
            <value>ATSC</value>
            <value>1080p30</value>
            <value>1080p24</value>
            <value>1080i60</value>
         </valueList>
      </dependentField>
      <minCount>1</minCount>
      <minListSize>1</minListSize>
      <maxListSize>11</maxListSize>
      <defaultValue>AUTO</defaultValue>
   </allowedValueDescriptor>
</field>
<allowedValue>720p60</allowedValue>
<allowedValue>720p30</allowedValue>
<allowedValue>720p24</allowedValue>
<allowedValue>480p60</allowedValue>
<allowedValue>480p30</allowedValue>
<allowedValue>480p24</allowedValue>
<allowedValue>480i60</allowedValue>
<allowedValue>AUTO</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
<dependentField>
  <name>srs:desiredRecordQuality@type</name>
  <valueList>
    <value>QLEVEL</value>
  </valueList>
</dependentField>
<minCount>1</minCount>
<minListSize>1</minListSize>
<maxListSize>4</maxListSize>
<allowedValueList>
  <allowedValue>Q1</allowedValue>
  <allowedValue>Q2</allowedValue>
  <allowedValue>Q3</allowedValue>
  <allowedValue>AUTO</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
</field>

<field>
  <name>srs:desiredRecordQuality@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField defaultDependency="1">
      <name>srs:desiredRecordQuality</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <defaultValue>DEFAULT</defaultValue>
    <allowedValueList>
      <allowedValue>DEFAULT</allowedValue>
      <allowedValue>ATSC</allowedValue>
      <allowedValue>QLEVEL</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledCDSObjectId</name>
  <dataType maxSize="1024">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE_DIRECT.CDSEPG</value>
        <value>OBJECT.RECORDSCHEDULE_DIRECT.CDSNONEPG</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
  </allowedValueDescriptor>
</field>
<field>
  <name>srs:scheduledChannelID</name>
  <dataType maxSize="256">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <allowAny/>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledChannelID@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:scheduledChannelID</name>
      <anyValue/>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>ANALOG</allowedValue>
      <allowedValue>DIGITAL</allowedValue>
      <allowedValue>FREQUENCY</allowedValue>
      <allowedValue>SI</allowedValue>
      <allowedValue>LINE</allowedValue>
      <allowedValue>NETWORK</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledChannelID@distriNetworkName</name>
  <dataType maxSize="32">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:scheduledChannelID</name>
      <anyValue/>
    </dependentField>
    <allowAny/>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledChannelID@distriNetworkID</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:scheduledChannelID</name>
      <anyValue/>
    </dependentField>
    <allowAny/>
  </allowedValueDescriptor>
</field>
<allowedValueDescriptor>
</field>

[field]
  <name>srs:scheduledStartDateTime</name>
  <dataType maxSize="64">xsd:string</dataType>
  <maxCountTotal>2</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
      </valueList>
      <minCount>1</minCount>
      <allowAny></allowAny>
    </allowedValueDescriptor>
  </dependentField>
  <minCount>1</minCount>
  <allowAny></allowAny>
</field>

[field]
  <name>srs:scheduledDuration</name>
  <dataType maxSize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
      </valueList>
      <minCount>1</minCount>
      <allowAny></allowAny>
    </allowedValueDescriptor>
  </dependentField>
  <minCount>1</minCount>
  <allowAny></allowAny>
</field>

[field]
  <name>srs:scheduledProgramCode</name>
  <dataType maxSize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.PROGRAMCODE</value>
      </valueList>
      <minCount>1</minCount>
      <allowAny></allowAny>
    </allowedValueDescriptor>
  </dependentField>
  <minCount>1</minCount>
  <allowAny></allowAny>
</field>

[field]
  <name>srs:scheduledProgramCode@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <name>srs:scheduledProgramCode</name>
    <anyValue></anyValue>
</field>
<field>
  <name>srs:matchingName</name>
  <dataType maxSize="128">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
      </valueList>
    </dependentField>
    <minCount>1</minCount>
    <allowAny/></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingName@type</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchingName</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>PROGRAM</allowedValue>
      <allowedValue>SERIES</allowedValue>
    </allowedValueList>
    <defaultValue>1</defaultValue>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingName@subStringMatch</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchingName</name>
      <anyValue></anyValue>
    </dependentField>
    <defaultValue>1</defaultValue>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingID</name>
  <dataType maxSize="256">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
      </valueList>
    </dependentField>
  </allowedValueDescriptor>
</field>
<field>
    <name>srs:matchingID@type</name>
    <dataType maxSize="16">xsd:string</dataType>
    <allowedValueDescriptor>
        <dependentField>
            <name>srs:matchingID</name>
            <anyValue></anyValue>
        </dependentField>
        <minCount>1</minCount>
        <allowedValueList>
            <allowedValue>SI_PROGRAMID</allowedValue>
            <allowedValue>SI_SERIESID</allowedValue>
        </allowedValueList>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:matchingChannelID</name>
    <dataType maxSize="256">xsd:string</dataType>
    <allowedValueDescriptor>
        <dependentField>
            <name>srs:class</name>
            <valueList>
                <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
            </valueList>
        </dependentField>
        <allowAny></allowAny>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:matchingChannelID@type</name>
    <dataType maxSize="16">xsd:string</dataType>
    <allowedValueDescriptor>
        <dependentField>
            <name>srs:matchingChannelID</name>
            <anyValue></anyValue>
        </dependentField>
        <minCount>1</minCount>
        <allowedValueList>
            <allowedValue>ANALOG</allowedValue>
            <allowedValue>DIGITAL</allowedValue>
            <allowedValue>FREQUENCY</allowedValue>
            <allowedValue>SI</allowedValue>
            <allowedValue>LINE</allowedValue>
            <allowedValue>NETWORK</allowedValue>
        </allowedValueList>
    </allowedValueDescriptor>
</field>

<field>
    <name>srs:matchingChannelID@distriNetworkName</name>
    <dataType maxSize="32">xsd:string</dataType>
    <allowedValueDescriptor>
        <dependentField>
            <name>srs:class</name>
            <valueList>
                <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
            </valueList>
        </dependentField>
        <allowAny></allowAny>
    </allowedValueDescriptor>
</field>
<field>
  <name>srs:matchingChannelID@distriNetworkID</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchingChannelID</name>
      <anyValue></anyValue>
    </dependentField>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingStartDateTimeRange</name>
  <dataType maxSize="64">xsd:string</dataType>
  <maxCountTotal>3</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
      </valueList>
    </dependentField>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingDurationRange</name>
  <dataType maxSize="16">xsd:string</dataType>
  <maxCountTotal>4</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
      </valueList>
    </dependentField>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingRatingLimit</name>
  <dataType maxSize="16">xsd:string</dataType>
  <maxCountTotal>2</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
      </valueList>
    </dependentField>
  </allowedValueDescriptor>
</field>
<value><u>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</u></value>
</dependentField>
</dependentField>
<dependentField>
<name><u>srs:matchingRatingLimit@type</u></name>
<valueList>
  <value><u>MPAA.ORG</u></value>
</valueList>
</dependentField>
<allowedValueList>
  <allowedValue>G</allowedValue>
  <allowedValue>PG</allowedValue>
  <allowedValue>PG-13</allowedValue>
  <allowedValue>R</allowedValue>
  <allowedValue>NC-17</allowedValue>
  <allowedValue>NR</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
<allowedValueDescriptor>
<dependentField>
<name><u>srs:matchingRatingLimit@type</u></name>
<valueList>
  <value><u>RIAA.ORG</u></value>
</valueList>
</dependentField>
<allowedValueList>
  <allowedValue>PA-EC</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
<allowedValueDescriptor>
<dependentField>
<name><u>srs:matchingRatingLimit@type</u></name>
<valueList>
  <value><u>ESRB.ORG</u></value>
</valueList>
</dependentField>
<allowedValueList>
  <allowedValue>EC</allowedValue>
  <allowedValue>E</allowedValue>
  <allowedValue>E10+</allowedValue>
  <allowedValue>T</allowedValue>
  <allowedValue>AO</allowedValue>
  <allowedValue>RP</allowedValue>
</allowedValueList>
</allowedValueDescriptor>
<field>
  <name>srs:matchingRatingLimit@type</name>
  <dataType maxSize="32">xsd:string</dataType>
  <maxCountTotal>2</maxCountTotal>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:matchingRatingLimit</name>
      <anyValue></anyValue>
    </dependentField>
    <minCount>1</minCount>
    <allowedValueList>
      <allowedValue>MPAA.ORG</allowedValue>
      <allowedValue>RIAA.ORG</allowedValue>
      <allowedValue>ESRB.ORG</allowedValue>
      <allowedValue>TVGUIDELINES.ORG</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:matchingEpisodeType</name>
  <dataType maxSize="8">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
      </valueList>
    </dependentField>
    <allowedValueList>
      <allowedValue>ALL</allowedValue>
      <allowedValue>FIRST_RUN</allowedValue>
    </allowedValueList>
  </allowedValueDescriptor>
</field>
<field>
  <name>srs:totalDesiredRecordTasks</name>
  <dataType>xsd:unsignedInt</dataType>
  <allowedValueDescriptor>
    <defaultValue>1</defaultValue>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledStartDateTimeAdjust</name>
  <dataType maxSize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <defaultValue>+P00:00:00</defaultValue>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:scheduledDurationAdjust</name>
  <dataType maxSize="16">xsd:string</dataType>
  <allowedValueDescriptor>
    <defaultValue>+P00:00:00</defaultValue>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:activePeriod</name>
  <dataType maxSize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.MANUAL</value>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSNONEPG</value>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
      </valueList>
      <defaultValue>NOW/INFINITY</defaultValue>
      <allowAny></allowAny>
    </dependentField>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:durationLimit</name>
  <dataType maxSize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECORDSCHEDULE.DIRECT.CDSEPG</value>
        <value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTNAME</value>
      </valueList>
    </dependentField>
  </allowedValueDescriptor>
</field>
<value>OBJECT.RECORDSCHEDULE.QUERY.CONTENTID</value>
</valueList>
</dependentField>
<defaultValue>INFINITY</defaultValue>
<allowAny/>
</allowedValueDescriptor>
</field>

[field]
<name>srs:durationLimit@effect</name>
<dataType maxSize="8">xsd:string</dataType>
<allowedValueDescriptor>
<dependentField defaultDependency="1">
<name>srs:durationLimit</name>
<anyValue/>
</dependentField>
<defaultValue>FIRST</defaultValue>
<allowedValueList>
<allowedValue>FIRST</allowedValue>
<allowedValue>LAST</allowedValue>
<allowedValue>Skip</allowedValue>
</allowedValueList>

</allowedValueDescriptor>
</field>

[field]
<name>srs:channelMigration</name>
<dataType>xsd:boolean</dataType>
<allowedValueDescriptor>
<dependentField>
<name>srs:class</name>
</dependentField>
<defaultValue>1</defaultValue>
<allowAny/>
</allowedValueDescriptor>
</field>

[field]
<name>srs:timeMigration</name>
<dataType>xsd:boolean</dataType>
<allowedValueDescriptor>
<dependentField>
<name>srs:class</name>
</dependentField>
<defaultValue>1</defaultValue>
<allowAny/>
</allowedValueDescriptor>
</field>
<field>
  <name>srs:allowDuplicates</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField>
      <name>srs:class</name>
      <valueList>
        <value>OBJECT.RECRODSCHEDULE.QUERY.CONTENTNAME</value>
        <value>OBJECT.RECRODSCHEDULE.QUERY.CONTENTID</value>
      </valueList>
    </dependentField>
    <defaultValue>1</defaultValue>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:persistedRecordings</name>
  <dataType>xsd:unsignedInt</dataType>
  <allowedValueDescriptor>
    <defaultValue>0</defaultValue>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:persistedRecordings@latest</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField defaultDependency="1">
      <name>srs:persistedRecordings</name>
      <anyValue></anyValue>
    </dependentField>
    <defaultValue>1</defaultValue>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:persistedRecordings@preAllocation</name>
  <dataType>xsd:boolean</dataType>
  <allowedValueDescriptor>
    <dependentField defaultDependency="1">
      <name>srs:persistedRecordings</name>
      <anyValue></anyValue>
    </dependentField>
    <defaultValue>0</defaultValue>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>

<field>
  <name>srs:persistedRecordings@storedLifetime</name>
  <dataType maxsize="64">xsd:string</dataType>
  <allowedValueDescriptor>
    <dependentField defaultDependency="1">
      <name>srs:persistedRecordings</name>
      <anyValue></anyValue>
    </dependentField>
    <defaultValue>ANY</defaultValue>
    <allowAny></allowAny>
  </allowedValueDescriptor>
</field>
</allowedValueDescriptor>
</field>

</fieldTable>
</AVDT>