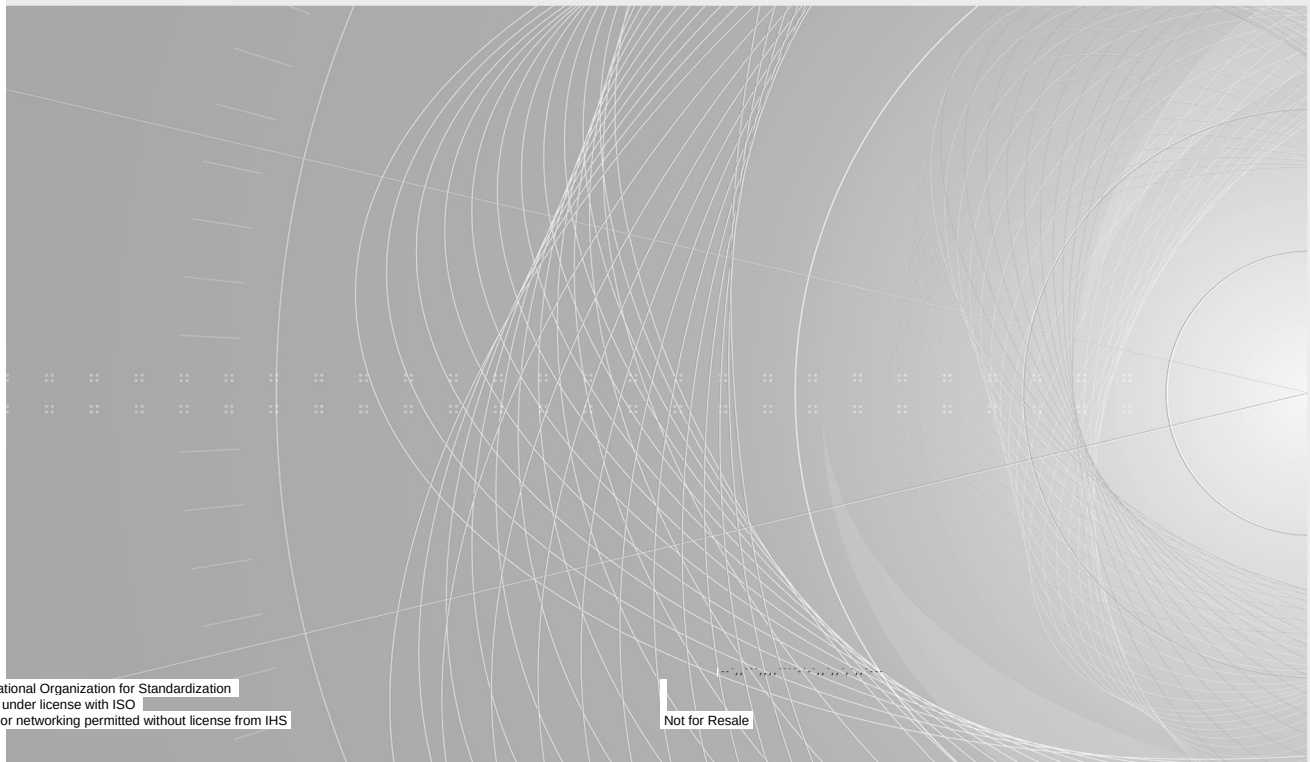
ISO/IEC 29341-4-11

Edition 1.0 2008-11

INTERNATIONAL STANDARD

Information technology – UPnP Device Architecture – Part 4-11: Audio Video Device Control Protocol – Level 2 – Connection Manager Service



ISO/IEC 29341-4-11:2008(E)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2008 ISO/IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester.

If you have any questions about ISO/IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub

The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ISO/IEC 29341-4-11

Edition 1.0 2008-11

INTERNATIONAL STANDARD

Information technology – UPnP Device Architecture – Part 4-11: Audio Video Device Control Protocol – Level 2 – Connection Manager Service

INTERNATIONAL ELECTROTECHNICAL COMMISSION

PRICE CODE S

ICS 35.200

ISBN 2-8318-1006-6

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

CONTENTS

FOREWORD 4
ORIGINAL UPNP DOCUMENTS (informative) 6
1 Overview and Scope 8
1.1 Introduction 8
1.2 Notation 8
1.2.1 Data Types 8
1.2.2 Strings Embedded in Other Strings 9
1.2.3 Extended Backus-Naur Form 9
1.3 Derived Data Types 10
1.3.1 Comma Separated Value (CSV) Lists 10
1.4 Management of XML Namespaces in Standardized DCPs 11
1.4.1 Namespace Prefix Requirements 14
1.4.2 Namespace Names, Namespace Versioning and Schema Versioning 15
1.4.3 Namespace Usage Examples 16
1.5 Vendor-defined Extensions 16
1.6 References 17
2 Service Modeling Definitions 20
2.1 ServiceType 20
2.2 State Variables 20
2.2.1 *SourceProtocolInfo* 21
2.2.2 *SinkProtocolInfo* 21
2.2.3 *CurrentConnectionIDs* 21
2.2.4 *A_ARG_TYPE_ConnectionStatus* 22
2.2.5 *A_ARG_TYPE_ConnectionManager* 22
2.2.6 *A_ARG_TYPE_Direction* 22
2.2.7 *A_ARG_TYPE_ProtocolInfo* 22
2.2.8 *A_ARG_TYPE_ConnectionID* 22
2.2.9 *A_ARG_TYPE_AVTransportID* 22
2.2.10 *A_ARG_TYPE_RcsID* 22
2.3 Eventing and Moderation 23
2.4 Actions 24
2.4.1 *GetProtocolInfo()* 25
2.4.2 *PrepareForConnection()* 25
2.4.3 *ConnectionComplete()* 28
2.4.4 *GetCurrentConnectionIDs()* 29
2.4.5 *GetCurrentConnectionInfo()* 29
2.4.6 Common Error Codes 31
2.5 Theory of Operation 32
2.5.1 Purpose 32
2.5.2 *ProtocolInfo* Concept 32
2.5.3 Typical Control Point Operations 37
2.5.4 Relation to Devices without ConnectionManagers 38
2.5.5 *PrepareForConnection()* and *ConnectionComplete()* 38
3 XML Service Description 42
4 Test 46

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A (normative) **Protocol Specifics** **47**
A.1 Application to HTTP Streaming 47
A.1.1 *ProtocolInfo* Definition 47
A.1.2 Implementation of *PrepareForConnection()* 47
A.1.3 Implementation of *ConnectionComplete()* 47
A.1.4 Automatic Connection Cleanup 47
A.2 Application to RTSP/RTP/UDP Streaming 48
A.2.1 *ProtocolInfo* Definition 48
A.2.2 Implementation of *PrepareForConnection()* 48
A.2.3 Implementation of *ConnectionComplete()* 48
A.2.4 Automatic Connection Cleanup 48
A.3 Application to Device-Internal Streaming 49
A.4 Application to IEC61883 Streaming 49
A.4.1 *ProtocolInfo* Definition 49
A.4.2 Implementation of *PrepareForConnection()* 50
A.4.3 Implementation of *ConnectionComplete()* 51
A.4.4 Automatic Connection Cleanup 51
A.5 Application to Vendor-specific Streaming 52

LIST OF TABLES

Table 1-1: EBNF Operators 9
Table 1-2: CSV Examples 11
Table 1-3: Namespace Definitions 13
Table 1-4: Schema-related Information 14
Table 1-5: Default Namespaces for the AV Specifications 15
Table 2-6: State Variables 20
Table 2-7: Event Moderation 23
Table 2-8: Actions 24
Table 2-9: Arguments for *GetProtocolInfo()* 25
Table 2-10: Arguments for *PrepareForConnection()* 26
Table 2-11: Error Codes for *PrepareForConnection()* 27
Table 2-12: Arguments for *ConnectionComplete()* 28
Table 2-13: Error Codes for *ConnectionComplete()* 29
Table 2-14: Arguments for *GetCurrentConnectionIDs()* 29
Table 2-15: Error Codes for *GetCurrentConnectionIDs()* 29
Table 2-16: Arguments for *GetCurrentConnectionInfo()* 30
Table 2-17: Error Codes for *GetCurrentConnectionInfo()* 30
Table 2-18: Common Error Codes 31
Table 2-19: Defined Protocols and their associated *ProtocolInfo* Values 33
Table A-1: <contentFormat> for Protocol IEC61883 50

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

INFORMATION TECHNOLOGY – UPNP DEVICE ARCHITECTURE –

Part 4-11: Audio Video Device Control Protocol – Level 2 – Connection Manager Service

FOREWORD

1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards. Their preparation is entrusted to technical committees; any ISO and IEC member body interested in the subject dealt with may participate in this preparatory work. International governmental and non-governmental organizations liaising with ISO and IEC also participate in this preparation.

2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

3) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO member bodies.

4) IEC, ISO and ISO/IEC publications have the form of recommendations for international use and are accepted by IEC and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

5) In order to promote international uniformity, IEC and ISO member bodies undertake to apply IEC, ISO and ISO/IEC publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO/IEC publication and the corresponding national or regional publication should be clearly indicated in the latter.

6) ISO and IEC provide no marking procedure to indicate their approval and cannot be rendered responsible for any equipment declared to be in conformity with an ISO/IEC publication.

7) All users should ensure that they have the latest edition of this publication.

8) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.

9) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

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

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

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

Information may be obtained from:

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

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

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

Information may be obtained from:

Microsoft Corporation
One Microsoft Way
USA – Redmond WA 98052

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

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

Information may be obtained from:

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

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

Information may be obtained from:

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

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

ISO/IEC 29341-4-11 was prepared by UPnP Implementers Corporation and adopted, under the PAS procedure, by joint technical committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

The list of all currently available parts of the ISO/IEC 29341 series, under the general title *Universal plug and play (UPnP) architecture*, can be found on the IEC web site.

This International Standard has been approved by vote of the member bodies, and the voting results may be obtained from the address given on the second title page.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ORIGINAL UPNP DOCUMENTS
(informative)

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

UPnP Document Title	**ISO/IEC 29341 Part**
UPnP Device Architecture 1.0	ISO/IEC 29341-1
UPnP Basic:1 Device	ISO/IEC 29341-2
UPnP AV Architecture:1	ISO/IEC 29341-3-1
UPnP MediaRenderer:1 Device	ISO/IEC 29341-3-2
UPnP MediaServer:1 Device	ISO/IEC 29341-3-3
UPnP AVTransport:1 Service	ISO/IEC 29341-3-10
UPnP ConnectionManager:1 Service	ISO/IEC 29341-3-11
UPnP ContentDirectory:1 Service	ISO/IEC 29341-3-12
UPnP RenderingControl:1 Service	ISO/IEC 29341-3-13
UPnP MediaRenderer:2 Device	ISO/IEC 29341-4-2
UPnP MediaServer:2 Device	ISO/IEC 29341-4-3
UPnP AV Datastructure Template:1	ISO/IEC 29341-4-4
UPnP AVTransport:2 Service	ISO/IEC 29341-4-10
UPnP ConnectionManager:2 Service	ISO/IEC 29341-4-11
UPnP ContentDirectory:2 Service	ISO/IEC 29341-4-12
UPnP RenderingControl:2 Service	ISO/IEC 29341-4-13
UPnP ScheduledRecording:1	ISO/IEC 29341-4-14
UPnP DigitalSecurityCamera:1 Device	ISO/IEC 29341-5-1
UPnP DigitalSecurityCameraMotionImage:1 Service	ISO/IEC 29341-5-10
UPnP DigitalSecurityCameraSettings:1 Service	ISO/IEC 29341-5-11
UPnP DigitalSecurityCameraStillImage:1 Service	ISO/IEC 29341-5-12
UPnP HVAC_System:1 Device	ISO/IEC 29341-6-1
UPnP HVAC_ZoneThermostat:1 Device	ISO/IEC 29341-6-2
UPnP ControlValve:1 Service	ISO/IEC 29341-6-10
UPnP HVAC_FanOperatingMode:1 Service	ISO/IEC 29341-6-11
UPnP FanSpeed:1 Service	ISO/IEC 29341-6-12
UPnP HouseStatus:1 Service	ISO/IEC 29341-6-13
UPnP HVAC_SetpointSchedule:1 Service	ISO/IEC 29341-6-14
UPnP TemperatureSensor:1 Service	ISO/IEC 29341-6-15
UPnP TemperatureSetpoint:1 Service	ISO/IEC 29341-6-16
UPnP HVAC_UserOperatingMode:1 Service	ISO/IEC 29341-6-17
UPnP BinaryLight:1 Device	ISO/IEC 29341-7-1
UPnP DimmableLight:1 Device	ISO/IEC 29341-7-2
UPnP Dimming:1 Service	ISO/IEC 29341-7-10
UPnP SwitchPower:1 Service	ISO/IEC 29341-7-11
UPnP InternetGatewayDevice:1 Device	ISO/IEC 29341-8-1
UPnP LANDevice:1 Device	ISO/IEC 29341-8-2
UPnP WANDevice:1 Device	ISO/IEC 29341-8-3
UPnP WANConnectionDevice:1 Device	ISO/IEC 29341-8-4
UPnP WLANAccessPointDevice:1 Device	ISO/IEC 29341-8-5
UPnP LANHostConfigManagement:1 Service	ISO/IEC 29341-8-10
UPnP Layer3Forwarding:1 Service	ISO/IEC 29341-8-11
UPnP LinkAuthentication:1 Service	ISO/IEC 29341-8-12
UPnP RadiusClient:1 Service	ISO/IEC 29341-8-13
UPnP WANCableLinkConfig:1 Service	ISO/IEC 29341-8-14
UPnP WANCommonInterfaceConfig:1 Service	ISO/IEC 29341-8-15
UPnP WANDSLLinkConfig:1 Service	ISO/IEC 29341-8-16
UPnP WANEthernetLinkConfig:1 Service	ISO/IEC 29341-8-17
UPnP WANIPConnection:1 Service	ISO/IEC 29341-8-18
UPnP WANPOTSLinkConfig:1 Service	ISO/IEC 29341-8-19
UPnP WANPPPConnection:1 Service	ISO/IEC 29341-8-20
UPnP WLANConfiguration:1 Service	ISO/IEC 29341-8-21
UPnP Printer:1 Device	ISO/IEC 29341-9-1
UPnP Scanner:1.0 Device	ISO/IEC 29341-9-2
UPnP ExternalActivity:1 Service	ISO/IEC 29341-9-10
UPnP Feeder:1.0 Service	ISO/IEC 29341-9-11
UPnP PrintBasic:1 Service	ISO/IEC 29341-9-12
UPnP Scan:1 Service	ISO/IEC 29341-9-13
UPnP QoS Architecture:1.0	ISO/IEC 29341-10-1
UPnP QosDevice:1 Service	ISO/IEC 29341-10-10
UPnP QosManager:1 Service	ISO/IEC 29341-10-11
UPnP QosPolicyHolder:1 Service	ISO/IEC 29341-10-12
UPnP QoS Architecture:2	ISO/IEC 29341-11-1
UPnP QOS v2 Schema Files	ISO/IEC 29341-11-2

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

UPnP Document Title	ISO/IEC 29341 Part
UPnP QosDevice:2 Service	ISO/IEC 29341-11-10
UPnP QosManager:2 Service	ISO/IEC 29341-11-11
UPnP QosPolicyHolder:2 Service	ISO/IEC 29341-11-12
UPnP RemoteUIClientDevice:1 Device	ISO/IEC 29341-12-1
UPnP RemoteUIServerDevice:1 Device	ISO/IEC 29341-12-2
UPnP RemoteUIClient:1 Service	ISO/IEC 29341-12-10
UPnP RemoteUIServer:1 Service	ISO/IEC 29341-12-11
UPnP DeviceSecurity:1 Service	ISO/IEC 29341-13-10
UPnP SecurityConsole:1 Service	ISO/IEC 29341-13-11

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

1 Overview and Scope

1.1 Introduction

This service definition is compliant with the UPnP Device Architecture version 1.0.

This service-type enables modeling of streaming capabilities of A/V devices, and binding of those capabilities between devices. Each device that is able to send or receive a stream according to the UPnP AV Architecture will have 1 instance of the ConnectionManager service. This service provides a mechanism for control points to:

1. Perform capability matching between source/server devices and sink/renderer devices,
2. Find information about currently ongoing transfers in the network,
3. Setup and teardown connections between devices (when required by the streaming protocol).

The ConnectionManager service is generic enough to properly abstract different kinds of streaming mechanisms, such as HTTP-based streaming, RTSP/RTP-based and 1394-based streaming.

The ConnectionManager enables control points to abstract from physical media interconnect technology when making connections. The term 'stream' used in this service template refers to both analog and digital data transfer.

1.2 Notation

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

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

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

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

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

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

  These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.
- Strings that are to be taken literally are enclosed in "double quotes".
- Words that are emphasized are printed in *italic*.
- Keywords that are defined by the UPnP AV Working Committee are printed using the *forum* character style.
- Keywords that are defined by the UPnP Device Architecture are printed using the **arch** character style.
- A double colon delimiter, "::", signifies a hierarchical parent-child (parent::child) relationship between the two objects separated by the double colon. This delimiter is used in multiple contexts, for example: Service::Action(), Action()::Argument, parentProperty::childProperty.

1.2.1 Data Types

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

For UPnP Device Architecture defined Boolean data types, it is strongly RECOMMENDED to use the value "**0**" for false, and the value "**1**" for true. However, when used as input arguments, the values "**false**", "**no**", "**true**",

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

"**yes**" may also be encountered and MUST be accepted. Nevertheless, it is strongly RECOMMENDED that all state variables and output arguments be represented as "**0**" and "**1**".

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

1.2.2 Strings Embedded in Other Strings

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

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

1.2.3 Extended Backus-Naur Form

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

1.2.3.1 Typographic conventions for EBNF

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

Table 1-1: EBNF Operators

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

1.3 Derived Data Types

This section defines a derived data type that is represented as a string data type with special syntax. This specification uses string data type definitions that originate from two different sources. The UPnP Device Architecture defined **string** data type is used to define state variable and action argument **string** data types. The XML Schema namespace is used to define property xsd:string data types. The following definition applies to both string data types.

1.3.1 Comma Separated Value (CSV) Lists

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

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

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 1-2: CSV Examples

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

1.4 Management of XML Namespaces in Standardized DCPs

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

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

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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

For the same reason that namespace prefixes are convenient in XML documents, it is convenient in specification text to refer to namespaces using a namespace prefix. Therefore, this specification declares a “standard” prefix for all XML namespaces used herein. In addition, this specification expands the scope where these prefixes have meaning, beyond a single XML document, to all of its text, XML examples, and certain string-valued properties. This expansion of scope *does not* supercede XML rules for usage in documents, it only augments and complements them in important contexts that are out-of-scope for the XML specifications.

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

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

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 1-3: Namespace Definitions

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1-4: Schema-related Information

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

1.4.1 Namespace Prefix Requirements

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

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

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Table 1-5: Default Namespaces for the AV Specifications

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

1.4.2 Namespace Names, Namespace Versioning and Schema Versioning

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

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

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

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

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

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

Namespaces are referenced in both schema and XML instance documents by namespace name. The namespace name appears as the value of an `xmlns` attribute. The `xmlns` attribute also declares a namespace prefix that will be used to qualify names from each namespace. Schemas are referenced in both schema and XML instance documents by URI in the `schemaLocation` attribute. See section 1.4.3, "Namespace Usage Examples" . Two different forms of URI are available, each with a different meaning. All UPnP AV-defined schema URIs share a common base path of "http://www.upnp.org/schemas/av/". Each schema URI has two unique relative forms (see Table 1-4, "Schema-related Information"), according to which version of a namespace and its representative schema is of interest. The allowed relative URI forms are:

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

Usage rules for schema location URIs are as follows:

- All instance documents, whether generated by a service or a control point, MUST use Form 1.
- All UPnP AV published schemas that reference other UPnP AV schemas will also use Form 1.
- Validation of XML instance documents in UPnP AV systems potentially serves two purposes. The first is based on standard XML and XML Schema semantics: the document's creator asserts that the

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

document is syntactically correct with respect to the referenced schema. The receiving processor can confirm this with a validating parser that uses the referenced schema(s). The second is based on UPnP AV namespace semantics. The receiving processor knows that the XML instance document is supposed to conform to one or more specific UPnP AV specifications. Since the second context is actually the more important context for instance document processing, the receiving processor MAY validate the instance document against any version of a schema that satisfies its needs in assessing the acceptability of the received instance document.

1.4.3 Namespace Usage Examples

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

Example 1:

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

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
 xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
   urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite-v2-20060531.xsd
   urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp-v2-20061231.xsd">
   <item id="18" parentID="13" restricted="0">
      ...
   </item>
</DIDL-Lite>
```

Example 2:

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

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

1.5 Vendor-defined Extensions

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

1.6 References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[HTTP/1.1] – *HyperText Transport Protocol – HTTP/1.1*, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, June 1999.
Available at: http://www.ietf.org/rfc/rfc2616.txt.

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

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

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

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

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

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

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

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

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

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

[RFC 2119] – *IETF RFC 2119, Key words for use in RFCs to Indicate Requirement Levels*, S. Bradner, 1997.
Available at: http://www.faqs.org/rfcs/rfc2119.html.

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

[RFC 3339] – *IETF RFC 3339, Date and Time on the Internet: Timestamps*, G. Klyne, Clearswift Corporation, C. Newman, Sun Microsystems, July 2002.
Available at: http://www.ietf.org/rfc/rfc3339.txt.

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

[RTSP] – *IETF RFC 2326, Real Time Streaming Protocol (RTSP)*, H. Schulzrinne, A. Rao, R. Lanphier, April 1998.
Available at: http://www.ietf.org/rfc/rfc2326.txt.

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

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

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

[UAX 15] – *Unicode Standard Annex #15, Unicode Normalization Forms, version 4.1.0, revision 25*, M. Davis, M. Dürst, March 25, 2005.
Available at: http://www.unicode.org/reports/tr15/tr15-25.html.

[UNICODE COLLATION] – *Unicode Technical Standard #10, Unicode Collation Algorithm version 4.1.0*, M. Davis, K. Whistler, May 5, 2005.
Available at: http://www.unicode.org/reports/tr10/tr10-14.html.

[UPNP-XSD] – *XML Schema for ContentDirectory:2 Metadata*, UPnP Forum, May 31, 2006.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Available at: http://www.upnp.org/schemas/av/upnp-v2-20060531.xsd.
Latest version available at: http://www.upnp.org/schemas/av/upnp-v2.xsd.

[UTS 10] – *Unicode Technical Standard #10, Unicode Collation Algorithm, version 4.1.0, revision 14*, M. Davis, K. Whistler, May 5, 2005.
Available at: http://www.unicode.org/reports/tr10/tr10-14.html.

[UTS 35] – *Unicode Technical Standard #35, Locale Data Markup Language, version 1.3R1, revision 5*,.M. Davis, June 2, 2005.
Available at: http://www.unicode.org/reports/tr35/tr35-5.html.

[XML] – *Extensible Markup Language (XML) 1.0 (Third Edition)*, François Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, eds., W3C Recommendation, February 4, 2004.
Available at: http://www.w3.org/TR/2004/REC-xml-20040204.

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

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

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

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

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

[XMLSCHEMA-XSD] – *XML Schema for XML Schema*.
Available at: http://www.w3.org/2001/XMLSchema.xsd.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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:*ConnectionManager:2*

2.2 State Variables

Table 2-6: State Variables

Variable Name	R/O[1]	Data Type	Allowed Value	Default Value	Eng. Units
SourceProtocolInfo	*R*	**string**	CSV[2] (**string**)		
SinkProtocolInfo	*R*	**string**	CSV (**string**)		
CurrentConnectionIDs	*R*	**string**	CSV (**ui4**)		
A_ARG_TYPE_ ConnectionStatus	*R*	**string**	"*OK*", "*ContentFormatMismatch*", "*InsufficientBandwidth*", "*UnreliableChannel*", "*Unknown*", *vendor-defined*		
A_ARG_TYPE_ ConnectionManager	*R*	**string**			
A_ARG_TYPE_Direction	*R*	**string**	"*Output*", "*Input*"		
A_ARG_TYPE_ProtocolInfo	*R*	**string**			
A_ARG_TYPE_ConnectionID	*R*	**i4**			
A_ARG_TYPE_AVTransportID	*R*	**i4**			
A_ARG_TYPE_RcsID	*R*	**i4**			
Non-standard state variables implemented by a UPnP vendor go here	*X*	*TBD*	*TBD*	*TBD*	*TBD*

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

[2] CSV stands for Comma-Separated Value list. The type between brackets denotes the UPnP data type used for the elements inside the list. The CSV list concept is defined more formally in the ContentDirectory service template.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Note: State variables *A_ARG_TYPE_ConnectionID*, *A_ARG_TYPE_AVTransportID*, and *A_ARG_TYPE_RcsID* are specified as being of data type **i4** to accommodate the fact that some actions REQUIRE these arguments to contain the special value -1. This special value is used as a return value to indicate that the service is not implemented on the device or is not needed for a particular connection. It can also be used as an *InstanceID* input argument when the actual *InstanceID* value is not (yet) known or the service does not exist.

Action *GetCurrentConnectionIDs()* in this specification and all *InstanceID*s in other services (AVTransport service, RenderingControl service, …) use data type **ui4** to specify *InstanceID* variables. However, this does not present a problem since a valid *InstanceID* value is always a non-negative integer and is always generated through an argument that is of type **i4**, effectively limiting the valid range for any *InstanceID* to [0, 2^{31}-1]. This range always fits in the valid range of an argument of data type **ui4** (range is [0, 2^{32}-1]) so that an 'out-of-range' error will never occur during assignment.

2.2.1 *SourceProtocolInfo*

This state variable contains a Comma-Separated Value (CSV) list of information on protocols this ConnectionManager supports for 'sourcing' (sending) data, in its current state. (The content of the CSV list can change over time, for example due to local resource restrictions on the device.) Besides the traditional notion of the term 'protocol', the protocol-related information provided by the connection also contains other information such as supported content formats. See Section 2.5, "Theory of Operation" for a general discussion on the notion of protocol info. See the table in Section 2.5.2, "*ProtocolInfo* Concept" for specific allowed values for this state variable. If the device does not support sourcing data, this state variable MUST be set to the empty string.

During normal operation, a MediaServer SHOULD ensure that there is consistency between what is reported in the *SourceProtocolInfo* state variable and all the *res@protocolInfo* properties of the items that populate the ContentDirectory; that is: at least all protocols that are used by any of the content items SHOULD be enumerated in the *SourceProtocolInfo* state variable. (Wildcards ("*") can be used in *SourceProtocolInfo* to limit the number of entries in the CSV list.) Additional protocols that are supported by the MediaServer but are not currently used by any of the content items MAY also be listed.

Control points can use the *SourceProtocolInfo* CSV list to quickly find out what type of content this MediaServer is capable of serving to the network.

A MediaServer can report temporary unavailability of a protocol (for example, HTTP server temporarily down or codec not available) by removing the appropriate entries from the *SourceProtocolInfo* CSV list.

2.2.2 *SinkProtocolInfo*

This state variable contains a Comma-Separated Value (CSV) list of information on protocols this ConnectionManager supports for 'sinking' (receiving) data, in its current state. (The content of the CSV list can change over time, for example due to local resource restrictions on the device.) The format and allowed value list are the same as for the *SourceProtocolInfo* state variable. If the device does not support 'sinking' data, this state variable MUST be set to the empty string.

A MediaRenderer can report temporary unavailability of a protocol (for example, codec not available) by removing the appropriate entries from the *SinkProtocolInfo* CSV list.

2.2.3 *CurrentConnectionIDs*

Comma-Separated Value list of references to current active Connections. This list MAY change without explicit actions invoked by control points, for example by out-of-band cleanup or termination of finished connections.

If OPTIONAL action *PrepareForConnection()* is not implemented then this state variable MUST be set to "0", indicating that this ConnectionManager service only supports a single connection identified by ConnectionID = 0.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2.2.4 A_ARG_TYPE_ConnectionStatus

This state variable is introduced to provide type information for the *Status* argument in the *GetCurrentConnectionInfo()* action. This status MAY change dynamically due to changes in the network.

2.2.5 A_ARG_TYPE_ConnectionManager

This state variable is introduced to provide type information for the *PeerConnectionManager* argument in actions *PrepareForConnection()* and *GetCurrentConnectionInfo()*. A ConnectionManager reference takes the form of a **UDN**/**serviceId** pair (the slash is the delimiter). A control point can use UPnP discovery (SSDP) to obtain a ConnectionManager's description document from the UDN. Subsequently, the ConnectionManager's service description can be obtained by using the **serviceId** part of the reference.

2.2.6 A_ARG_TYPE_Direction

This state variable is introduced to provide type information for the *Direction* argument in action *PrepareForConnection()*.

2.2.7 A_ARG_TYPE_ProtocolInfo

This state variable is introduced to provide type information for the *RemoteProtocolInfo* argument in action *PrepareForConnection()* and the *ProtocolInfo* argument in action *GetCurrentConnectionInfo()*.

2.2.8 A_ARG_TYPE_ConnectionID

This state variable is introduced to provide type information for the *ConnectionID* argument in actions *PrepareForConnection()*, *ConnectionComplete()* and *GetCurrentConnectionInfo()*.

2.2.9 A_ARG_TYPE_AVTransportID

This state variable is introduced to provide type information for the *AVTransportID* argument in actions: *PrepareForConnection()* and *GetCurrentConnectionInfo()*. It identifies a logical instance of the AVTransport service associated with a Connection. See AVTransport:1 Specification, Section 2.5.6, "Supporting multiple virtual Transports" for more information.

2.2.10 A_ARG_TYPE_RcsID

This state variable is introduced to provide type information for the *RcsID* argument in actions *PrepareForConnection()* and *GetCurrentConnectionInfo()*. It identifies a logical instance of the Rendering Control service associated with a Connection. See RenderingControl:1 Specification, Section 1.2, "Multi-input Devices" and Section 2.5.1, "Multi-input Devices" for more information.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.3 Eventing and Moderation

Table 2-7: Event Moderation

Variable Name	Evented	Moderated Event	Max Event Rate[1]	Logical Combination	Min Delta per Event[2]
SourceProtocolInfo	*YES*	*NO*			
SinkProtocolInfo	*YES*	*NO*			
CurrentConnectionIDs	*YES*	*NO*			
Non-standard state variables implemented by a UPnP vendor go here	*TBD*	*TBD*	*TBD*	*TBD*	*TBD*

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

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2.4 Actions

Immediately following this table is detailed information about these actions, including short descriptions of the actions, the effects of the actions on state variables, and error codes defined by the actions.

Table 2-8: Actions

Name	R/O[1]
GetProtocolInfo()	*R*
PrepareForConnection()	*O*
ConnectionComplete()	*O*
GetCurrentConnectionIDs()	*R*
GetCurrentConnectionInfo()	*R*
Non-standard actions implemented by a UPnP vendor go here	*X*

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

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Note: Non-standard actions MUST be implemented in such a way that they do not interfere with the basic operation of the ConnectionManager service; that is: these actions MUST be optional and do not need to be invoked for the ConnectionManager service to operate normally.

2.4.1 *GetProtocolInfo()*

This action returns the protocol-related info that this ConnectionManager supports in its current state, as a Comma-Separated Value list of strings according to Table 2-19, "Defined Protocols and their associated *ProtocolInfo* ". Protocol-related information for 'sourcing' data is returned in the *Source* argument and protocol-related information for 'sinking' data is returned in the *Sink* argument. When this ConnectionManager resides in a device that only supports 'sourcing' of data, the *Sink* argument MUST return the empty string. Likewise, when this ConnectionManager resides in a device that only supports 'sinking' of data, the *Source* argument MUST return the empty string.

2.4.1.1 Arguments

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

Argument	Direction	relatedStateVariable
Source	*OUT*	*SourceProtocolInfo*
Sink	*OUT*	*SinkProtocolInfo*

2.4.1.2 Dependency on State

None.

2.4.1.3 Effect on State

None.

2.4.1.4 Errors

None

2.4.2 *PrepareForConnection()*

This OPTIONAL action is used to allow the device to prepare itself to connect to the network for the purposes of sending or receiving media content (for example, a video stream). *PrepareForConnection()* also allows the device to indicate whether or not it can establish a connection based on the current status of the device and/or the current conditions of the network.

The *RemoteProtocolInfo* input argument identifies the protocol, network, and format that MUST be used to transfer the content.

- If *PrepareForConnection()* is invoked on a MediaServer device, the *RemoteProtocolInfo* argument MUST be set to one of the *ProtocolInfo* entries from the CSV list obtained from the peer MediaRenderer device via the *GetProtocolInfo()* action. (See Section 2.5.2, "*ProtocolInfo* Concept" for details.) If the peer device does not implement *GetProtocolInfo()* (because it is not a MediaRenderer or not even a UPnP device), then the *RemoteProtocolInfo* argument MUST be set to one of the *ProtocolInfo* entries returned by the *GetProtocolInfo()* action on the local MediaServer device.
- If *PrepareForConnection()* is invoked on a MediaRenderer device, the *RemoteProtocolInfo* argument MUST be set to the value of the *protocolInfo* attribute of the content item (located in the ContentDirectory on the peer MediaServer device) that is going to be played. (See Section 2.5.2, "*ProtocolInfo* Concept" for details.) If the peer device does not implement a ContentDirectory service (because it is not a MediaServer or not even a UPnP device), then the *RemoteProtocolInfo* argument MUST be set to one of the *ProtocolInfo* entries returned by the *GetProtocolInfo()* action on the local MediaRenderer device.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The *ConnectionID* out argument is used to identify the connection that was prepared by the device in response to this invocation. The *ConnectionID* is a device-specific value and is NOT unique throughout the network. Therefore, the *ConnectionIDs* returned by the two end-points of the same connection will generally NOT be the same value. Refer to *GetCurrentConnectionIDs()* and/or the UPnP AV Device Architecture document for additional information. The *AVTransportID* and *RcsID* out arguments are used to identify the AVTransport and RenderingControl services that are associated with the connection. The returned values are the *InstanceIDs* that need to be used when invoking subsequent invocations of the AVTransport and RenderingControl Services. An *InstanceID* value of -1 indicates the device did not associate an AVTransport and/or RenderingControl service with this connection. The returned *ConnectionID*, *AVTransportID*, and *RcsID* become invalid when the device closes the connection. This will occur when *ConnectionComplete()* is invoked or any other time the device decides to close the connection (a.k.a auto-cleanup). Refer to *ConnectionComplete()* for additional information.

This action is marked OPTIONAL which means that each device manufacturer decides whether or not to implement it. Therefore, some devices will implement *PrepareForConnection()* while other devices will not. Since *PrepareForConnection()* allows a device to prepare itself to connect to the network, if a device has implemented that action, control points need to invoke *PrepareForConnection()* before attempting to stream any content; that is: before invoking *AVTransport::SetAVTransportURI()* (See Section 2.5.3, “Typical Control Point Operations”). Otherwise, the device may not operate correctly because it has not been properly configured. Additionally, control points need to invoke *PrepareForConnection()*, if implemented, so that the device can inform the control point, via an error code, that the device’s current operating environment is not able to accommodate the requested stream.

Once a connection has been prepared, it can be used to transfer several pieces of the content before calling *ConnectionComplete()* as long as each content item is compatible with the *RemoteProtocolInfo* argument that was passed into *PrepareForConnection()*; that is: each content item has the same media format as specified in *RemoteprotocolInfo*.

If a device does not implement *PrepareForConnection()*, it MUST only support a single connection at any time. This connection is implicitly assumed to be always present and is identified by ConnectionID = 0.

2.4.2.1 Arguments

Table 2-10: Arguments for *PrepareForConnection()*

Argument	Direction	relatedStateVariable
RemoteProtocolInfo	*IN*	*A_ARG_TYPE_ProtocolInfo*
PeerConnectionManager	*IN*	*A_ARG_TYPE_ConnectionManager*
PeerConnectionID	*IN*	*A_ARG_TYPE_ConnectionID*
Direction	*IN*	*A_ARG_TYPE_Direction*
ConnectionID	*OUT*	*A_ARG_TYPE_ConnectionID*
AVTransportID	*OUT*	*A_ARG_TYPE_AVTransportID*
RcsID	*OUT*	*A_ARG_TYPE_RcsID*

2.4.2.2 Dependency on State

None.

2.4.2.3 Effect on State

This action prepares the device to stream content to or from the specified peer ConnectionManager, according to the specified direction and protocol information. The *PeerConnectionManager* input argument identifies the ConnectionManager service on the other side of the connection. The *PeerConnectionID* input argument identifies the specific connection on that ConnectionManager

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

service. This information allows a control point to *link* a connection on device A to the corresponding connection on device B, via action *GetCurrentConnectionInfo()*. If the *PeerConnectionID* is not known by a control point (for example, this is the first of the two *PrepareForConnection()* actions), or the peer device doesn't implement *PrepareForConnection()* then this value MUST be set to reserved value -1.

This action returns a locally unique ID for the established Connection in the *ConnectionID* argument, and adds that *ConnectionID* to state variable *CurrentConnectionIDs*. This *ConnectionID* might be used by a control point to manually terminate the established Connection through (OPTIONAL) action *ConnectionComplete()*. It can also be used to retrieve information associated with the Connection via action *GetCurrentConnectionInfo()*. Value -1 is reserved, and MUST NOT be returned.

OPTIONALLY, this action returns a virtual *InstanceID* of a local AVTransport service in the *AVTransportID* argument. This *AVTransportID* MUST be passed as an input argument to the local AVTransport service action invocations. If the returned *AVTransportID* is -1 (reserved value), then there is no AVTransport service on this device that can be used to control the established connection. This is dependent on the 'push' or 'pull' nature of the streaming protocol.

OPTIONALLY, this action returns a virtual *InstanceID* of a local RenderingControl service in the *RcsID* argument. This *RcsID* MUST be passed as an input argument to the local RenderingControl service action invocations. If the returned *RcsID* is -1 (reserved value), then there is no RenderingControl service on this device, for example, because the device is a source device (MediaServer) rather than a sink device (MediaRenderer).

Due to local restrictions on the device running the ConnectionManager, state variables *SourceProtocolInfo* and/or *SinkProtocolInfo* MAY change (for example, certain physical ports on the device are not available anymore for new connections) as a result of this action.

2.4.2.4 Errors

Table 2-11: Error Codes for *PrepareForConnection()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
701	Incompatible protocol info	The connection cannot be established because the protocol info argument is incompatible.
702	Incompatible directions	The connection cannot be established because the directions of the involved ConnectionManagers (source/sink) are incompatible.
703	Insufficient network resources	The connection cannot be established because there are insufficient network resources (bandwidth, channels, etc.).
704	Local restrictions	The connection cannot be established because of local restrictions in the device. This might happen, for example, when physical resources on the device are already in use by other connections.
705	Access denied	The connection cannot be established because the client is not permitted to access the specified ConnectionManager.
707	Not in network	The connection cannot be established because the ConnectionManagers are not part of the same physical network.
708	Connection Table overflow	The connection cannot be established because the specified ConnectionManager has instantiated the maximum number of simultaneous connections it has room for in its internal data structures. Closing one connection will resolve the issue.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

errorCode	errorDescription	Description
709	Internal processing resources exceeded	The connection cannot be established because the device does not have sufficient internal processing resources to handle the new connection. Closing one or more connections on this device may resolve the issue.
710	Internal memory resources exceeded	The connection cannot be established because the device does not have sufficient internal memory resources to handle the new connection. Closing one or more connections on this device may resolve the issue.
711	Internal storage system capabilities exceeded	The connection cannot be established because the device does not have sufficient internal storage system capabilities to handle the new connection. Closing one or more connections on this device may resolve the issue.

2.4.3 *ConnectionComplete()*

This OPTIONAL action is used to inform the device that the specified connection, which was previously allocated by *PrepareForConnection()*, is no longer needed. Any resources that were allocated for that connection during *PrepareForConnection()* can be freed by the device at its discretion.

In some situations, *ConnectionComplete()* may never be invoked; for example, the control point spontaneously goes away. In order to prevent an unused connection from permanently consuming resources, the device SHOULD automatically cleanup unused connections. The process for determining when a connection SHOULD be automatically cleaned up is implementation dependent. For example, a device MAY decide to close a connection after the connection has been inactive for a certain period of time. Alternatively, a device MAY decide to close a connection when it needs to free the resources that are associated with the connection. See Section 2.5.5, "*PrepareForConnection()* and *ConnectionComplete()*" for additional information.

2.4.3.1 Arguments

Table 2-12: Arguments for *ConnectionComplete()*

Argument	Direction	relatedStateVariable
ConnectionID	*IN*	*A_ARG_TYPE_ConnectionID*

2.4.3.2 Dependency on State

None.

2.4.3.3 Effect on State

This action removes the connection referenced by argument *ConnectionID* by modifying state variable *CurrentConnectionIDs*, and (if necessary) performs any protocol-specific cleanup actions such as releasing network resources. See Annex A, "
(normative)

Protocol Specifics" for more details.

Due to local restrictions on the device running the ConnectionManager, state variables *SourceProtocolInfo* and/or *SinkProtocolInfo* MAY change (for example, certain physical ports on the device are freed up for new connections).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.4.3.4 Errors

Table 2-13: Error Codes for *ConnectionComplete()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
706	Invalid connection reference	The connection reference argument does not refer to a valid connection established by this service.

2.4.4 *GetCurrentConnectionIDs()*

This action returns a Comma-Separated Value list of *ConnectionID*s of currently ongoing Connections. A *ConnectionID* can be used to manually terminate a Connection via action *ConnectionComplete()*, or to retrieve additional information about the ongoing Connection via action *GetCurrentConnectionInfo()*.

If a device does not implement *PrepareForConnection()*, this action MUST return the single value "0".

2.4.4.1 Arguments

Table 2-14: Arguments for *GetCurrentConnectionIDs()*

Argument	Direction	relatedStateVariable
ConnectionIDs	*OUT*	*CurrentConnectionIDs*

2.4.4.2 Dependency on State

None.

2.4.4.3 Effect on State

None.

2.4.4.4 Errors

Table 2-15: Error Codes for *GetCurrentConnectionIDs()*

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

2.4.5 *GetCurrentConnectionInfo()*

This action returns associated information of the connection referred to by the *ConnectionID* input argument. The *AVTransportID* argument MAY be the reserved value -1 and the *PeerConnectionManager* argument MAY be the empty string in cases where the connection has been setup completely out of band, not involving a *PrepareForConnection()* action.

If OPTIONAL action *PrepareForConnection()* is not implemented then (limited) connection information can be retrieved for *ConnectionID* 0. The device MUST return all known information:

- *RcsID* MUST be 0 (a single instance of the RenderingControl service is implemented) or -1 (RenderingControl Service is not implemented)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

- *AVTransportID* MUST be 0 (a single instance of the AVTransport service is implemented) or -1 (AVTransport service is not implemented)
- *ProtocolInfo* MUST contain accurate information if it is known, otherwise it MUST be the empty string
- *PeerConnectionManager* MUST be the empty string
- *PeerConnectionID* MUST be -1
- *Direction* MUST be "*Input*" or "*Output*"
- *Status* MUST be "*OK*" or "*Unknown*"

2.4.5.1 Arguments

Table 2-16: Arguments for *GetCurrentConnectionInfo()*

Argument	Direction	relatedStateVariable
ConnectionID	*IN*	*A_ARG_TYPE_ConnectionID*
RcsID	*OUT*	*A_ARG_TYPE_RcsID*
AVTransportID	*OUT*	*A_ARG_TYPE_AVTransportID*
ProtocolInfo	*OUT*	*A_ARG_TYPE_ProtocolInfo*
PeerConnectionManager	*OUT*	*A_ARG_TYPE_ConnectionManager*
PeerConnectionID	*OUT*	*A_ARG_TYPE_ConnectionID*
Direction	*OUT*	*A_ARG_TYPE_Direction*
Status	*OUT*	*A_ARG_TYPE_ConnectionStatus*

2.4.5.2 Dependency on State

None.

2.4.5.3 Effect on State

None.

2.4.5.4 Errors

Table 2-17: Error Codes for *GetCurrentConnectionInfo()*

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
706	Invalid connection reference	The connection reference argument does not refer to a valid connection established by this service.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.4.6 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 2-18: Common Error Codes

errorCode	errorDescription	Description
400-499	TBD	See UPnP Device Architecture section on Control.
500-599	TBD	See UPnP Device Architecture section on Control.
600-699	TBD	See UPnP Device Architecture section on Control.
701	Incompatible protocol info	The connection cannot be established because the protocol info argument is incompatible.
702	Incompatible directions	The connection cannot be established because the directions of the involved ConnectionManagers (source/sink) are incompatible.
703	Insufficient network resources	The connection cannot be established because there are insufficient network resources (bandwidth, channels, etc.).
704	Local restrictions	The connection cannot be established because of local restrictions in the device. This might happen, for example, when physical resources on the device are already in use by other connections.
705	Access denied	The connection cannot be established because the client is not permitted to access the specified ConnectionManager.
706	Invalid connection reference	The connection reference argument does not refer to a valid connection established by this service.
707	Not in network	The connection cannot be established because the ConnectionManagers are not part of the same physical network.
708	Connection Table overflow	The connection cannot be established because the specified ConnectionManager has instantiated the maximum number of simultaneous connections it has room for in its internal data structures. Closing one connection will resolve the issue.
709	Internal processing resources exceeded	The connection cannot be established because the device does not have sufficient internal processing resources to handle the new connection. Closing one or more connections on this device may resolve the issue.
710	Internal memory resources exceeded	The connection cannot be established because the device does not have sufficient internal memory resources to handle the new connection. Closing one or more connections on this device may resolve the issue.
711	Internal storage system capabilities exceeded	The connection cannot be established because the device does not have sufficient internal storage system capabilities to handle the new connection. Closing one or more connections on this device may resolve the issue.

Note 1: The errorDescription field returned by an action does not necessarily contain human-readable text (for example, as indicated in the second column of the Error Code tables.) It may contain machine-readable information that provides more detailed information about the error. It is therefore not advisable for a control point to blindly display the errorDescription field contents to the user.

Note 2: 800-899 Error Codes are not permitted for standard actions. See UPnP Device Architecture section on Control for more details.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.5 Theory of Operation

2.5.1 Purpose

The purpose of the ConnectionManager is to enable control points to:

1. perform capability matching between source/server devices and sink/renderer devices. This involves both:
    a. content-format matching (for example, mp3 – mp3)
    b. transport (streaming) protocol matching (for example, http – http)
2. find information about currently ongoing streams in the network, for example:
    a. find the source device sending content to a given renderer device
    b. find the renderer devices served by a given source device or content resource
    c. find all streams going on in the network
3. setup and teardown connections between devices (when required by the streaming protocol)

2.5.2 *ProtocolInfo* Concept

While the UPnP Architecture describes, and prescribes, many aspects of devices that are required for a certain level of interoperability, it does not describe anything related to streaming between devices. The purpose of the ConnectionManager service is to make these aspects of devices explicit, so that control points are able to make intelligent choices, present intelligent user interfaces, and initiate (and terminate) streams between controlled devices via UPnP actions. UPnP-defined protocols are used to initiate (and terminate) the stream, even though they are not used to stream the actual data *packets*.

The ConnectionManager service defines the notion of *ProtocolInfo* as information needed by a control point in order to determine (a certain level of) compatibility between the streaming mechanisms of two UPnP controlled devices. For example, it contains the transport protocols supported by a device, for input or output, as well as other information such as the content formats (encodings) that can be sent, or received, via the transport protocols. Note that, while UPnP prescribes the use of HTTP for controlling devices via SOAP, it does NOT REQUIRE HTTP to be used for all kinds of (Audio and Video) streaming in a UPnP network.

In the context of this document, the term *ProtocolInfo* is used to describe a string formatted as:

<protocol>“:”<network>“:”<contentFormat>“:”<additionalInfo>

where each of the 4 elements MAY be a wildcard “*”. Control points can match *ProtocolInfo* by (protocol-independent) string comparison operations on the <protocol>, <network> and <contentFormat> elements, taking into account the “*” wildcard, which matches with anything. It is RECOMMENDED that control points perform string matching using case-insensitive comparison. However, devices are REQUIRED to provide the <protocol>, <network>, and <contentFormat> strings exactly as prescribed by this and other specifications.

When performing protocol matching, control points have basically three different sources for protocol information:

- The value of the *res@protocolInfo* property of the content item to be played, which is exposed by the ContentDirectory service.
- The Comma Separated Value list maintained in the *SinkProtocolInfo* state variable of the MediaRenderer device.
- The Comma Separated Value list maintained in the *SourceProtocolInfo* state variable of the MediaServer device.

Control points should match the content item’s *res@protocolInfo* property value to one of the *ProtocolInfo* entries in the MediaRenderer’s *SinkProtocolInfo* CSV list. In addition, a control point

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

may want to check whether the content item's *res@protocolInfo* property value matches one of the entries in the MediaServer's *SourceProtocolInfo* CSV list to ensure that the MediaServer is currently capable of serving this content item.

The <additionalInfo> part does not need to match between source and sink. Its purpose is to convey any additional information needed to set up the out of band stream (for example, 1394 addresses). The structure of this 4th field is described later in this section.

The following table summarizes how the protocol info strings are defined for the protocols currently standardized by the ConnectionManager service, as well as for vendor-defined protocols. Annex A, " **(normative)**

Protocol Specifics", provides a more detailed explanation per protocol.

Table 2-19: Defined Protocols and their associated *ProtocolInfo* Values

Protocol Name	protocol	network	contentFormat	additionalInfo	Ref.
HTTP GET	"*http-get*"	"*"[1]	MIME-type.	Vendor-defined, MAY be "*".	App. Annex A
RTSP/RTP/UDP	"*rtsp-rtp-udp*"	"*"[2]	Name of RTP payload type.	Vendor-defined, MAY be "*".	App. A.2
INTERNAL	"*internal*"	IP address of the device hosting the Connection-Manager.	Vendor-defined, MAY be "*".	Vendor-defined, MAY be "*".	App. A.3
IEC61883_EX1	"*iec61883_ex1*"	GUID of the 1394 bus Isochronous Resource Manager.	Name standardized by IEC61883.	upnp.org_GUID = <GUID-value>;<PCR-index>. See definitions below.	App. A.4
IEC61883	"*iec61883*"	GUID of the 1394 bus Isochronous Resource Manager.	Name standardized by IEC61883.	GUID and PCR index of the 1394 device. See IEC61883 exception below.	App. A.4
VENDOR	Registered ICANN domain name of vendor	Vendor-defined, MAY be "*".	Vendor-defined, MAY be "*".	Vendor-defined, MAY be "*".	App. A.5

[1] Since all devices supporting HTTP GET belong to the same IP network, the network does not need to be specified.

[2] Since all devices supporting RTSP/RTP/UDP belong to the same IP network, the network does not need to be specified.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2.5.2.1 4th Field – <additionalInfo>

Except for the IEC61883 protocol, the 4th field of the *ProtocolInfo* string contains either an asterisk character ("*") or a list of name-value pairs. An asterisk indicates that the 4th field does not contain any meaningful data and should be ignored. A list of name-value pairs indicates that there is some additional information beyond the first three fields. In this case, the 4th field MUST contain one or more name-value pairs (separated by a semi-colon ";") with each name-value pair having the following format:

<org-name>_<token-name>=<value>

where

<org-name> is the ICANN registered domain name of the organization that has defined the semantic of the name-value pair. For example, the UPnP Forum would use an <org-name> of "upnp.org". Case-insensitive comparison is used.

<token-name> identifies the additional data that is being defined. It consists of one or more alpha-numeric characters (i.e. 'a'-'z', 'A'-'Z', '0'-'9', '_') and MUST be unique within the context of the specified <org-name>. Case-insensitive comparison is used.

<value> is the value of the additional data. In addition to the escaping rule defined for attributes in [XML], the following rules also apply:

- All semi-colons (";") within <value> MUST be escaped with a backslash ("\"). This is necessary since a semi-colon is used to separate multiple name-value pair occurrences. For example, in order to represent a value of "yours;mine;ours", <value> MUST be set to "yours\;mine\;ours".
- All original backslash ("\") characters within <value> MUST be escaped with a (second) backslash ("\"). Obviously, backslash characters that have been added as an escape character are themselves not double escaped. For example, in order to represent a value of "yours\mine\ours", <value> MUST be set to "yours\\mine\\ours".

Multiple name-value pairs are separated by a semi-colon (";") and have the layout below. When multiple name-value pairs are specified, the order of occurrence of the name-value pair is not relevant. Additionally, the same value of <org-name> MAY occur multiple times and the same value of <token-name> MAY occur multiple times. However, each <org-name>_<token-name> combination MUST appear at most once within the list of name-value pairs contained by the 4th field. The following example shows three name value pairs: two of which are defined by the UPnP Forum and one of which is defined by a fictitious organization called "VendorA":

upnp.org_resolution=1080i;VendorA.com_resolution=super_high_quality;upnp.org_sample_rate=30FPS

As described before, the <additionalInfo> field may be used for any purpose and contains name-value pairs which the control point may or may not understand. Since the semantics of the unknown name-value pairs are unknown, it SHOULD ignore unknown name-value pairs and only known name-value pairs MAY be used during comparison.

2.5.2.2 IEC61883 Exception

When the IEC61883 protocol was first introduced into the specification, the structure of the 4th field was not yet defined. Therefore, the additional data that was needed for this protocol was simply placed directly in the 4th field without any higher-level constructs. In order to maintain compatibility with existing implementations of this specification, the definition of the 4th field for the IEC61883 protocol can not be changed in order to comply with the 4th field layout defined above. However, to eventually deprecate this non-conformant protocol designation, a new protocol designation has been defined for IEC61883 which does conform to the above layout. It has been named IEC61883_EX1 with the "_EX1" suffix indicating "Extension #1". All future implementations that support the IEC61883 protocol MUST use the new designator (IEC61883_EX1) as well as the original designator (IEC61883).

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.5.2.3 Formal EBNF for the 4th field

The formal EBNF for the 4th field is as follows:

```
4th-field                ::=   '*'|name-value-pair-list|IEC61883-
exception

name-value-pair-list ::=   name-value-pair (';' name-value-pair)*
name-value-pair      ::=   org-name '_' token-name '=' value

org-name                 ::=   (* ICANN registered domain name
                                  including the top-level domain suffix
                                  (e.g. ".com", ".org", ".netv, etc.) *)

token-name               ::=   ('a'-'z' | 'A'-'Z' | '0'-'9' | '_')+

value                    ::=   (unicode-char-except-backslash-semicolon|
                               escaped-backslash|escaped-semicolon)*

unicode-char-except-backslash-semicolon
                         ::=   (* any Unicode-4 character except a
                                  '\' or '; ' character *)

escaped-backslash        ::=   '\\'

escaped-semicolon        ::=   '\;'

IEC61883-exception       ::=   GUID-value ';' PCR-index

GUID-value               ::=   (* hex encoding of the device's
                                  IEC61883 node_vendor_id and chip_id
                                  (total of 64-bits) *)

PCR-index                ::=   (* zero-based integer index identifying
                                  the plug within the device *)
```

2.5.2.4 *ProtocolInfo* Conventions for Protected Content

2.5.2.4.1 3rd Field - MIME Type Format

MIME types for protected content resources appear in the third field, <contentFormat>, of the *ProtocolInfo* string. Since protected files may be described by both a content protection MIME type as well as a MIME type associated with the underlying media, the following convention for extended MIME types will be used for MIME types that describe protected resources:

<content_protection_MIME_type>;CONTENTFORMAT=<underlying_media_MIME_type>

Example 1. The following example describes a MPEG2-TS resource that is protected with DTCP-IP when streaming. The extended MIME type is:

application/x-dtcp1;CONTENTFORMAT=video/MP2T

The full resulting *ProtocolInfo* string additionally indicates that the stream is http-get:

ProtocolInfo = "http-get:*:application/x-dtcp1;CONTENTFORMAT=video/MP2T:*"

Finally, the form of the <res> element content is (per DTCP Volume 1 Supplement E Revision 1.0):

```
<res protocolInfo=
 "http-get:*:application/x-dtcp1;CONTENTFORMAT=video/MP2T:*"
 allowedUse="PLAY,5"
 validityStart="2004-05-30T14:30:00"
 validityEnd="2004-06-04T14:30:00">
   http://10.0.0.1:88/MyCollection/movie7829.mp2t?
   CONTENTPROTECTIONTYPE=DTCP1&amp;DTCP1HOST=1.2.3.4&amp;DTCP1PORT
=97
</res>
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Example 2. The following example extended MIME type describes a MPEG2-PS resource that is delivered as an OMA DCF file:

"application/vnd.oma.drm.dcf;CONTENTFORMAT=video/MP2P"

The resulting *ProtocolInfo* string containing the extended MIME type additionally indicates that the file is transferred with http-get:

ProtocolInfo = "http-get:*:application/vnd.oma.drm.dcf;CONTENTFORMAT=video/MP2P:*"

Finally, the form of the <res> element content is as follows:

```
<res protocolInfo=
"http-
get:*:application/vnd.oma.drm.dcf;CONTENTFORMAT=video/MP2P:*"
 allowedUse="PLAY,5"
 validityStart="2004-05-30T14:30:00"
 validityEnd="2004-06-04T14:30:00">
   http://10.0.0.1:88/MyCollection/movie8126.dcf
</res>
```

Example 3. The following example describes a MPEG2-PS resource that is delivered as an OMA DCF file and can be exported to a DTCP content protection system:

The rights object of the content includes the permissions for enabling the translation into the new DRM system and will be represented as two separate <res> elements representing the same content:

```
<res protocolInfo=
 "http-
get:*:application/vnd.oma.drm.dcf;CONTENTFORMAT=video/MP2P:*"
 allowedUse="PLAY,5"
 validityStart="2004-05-30T14:30:00"
 validityEnd="2004-06-04T14:30:00">
   http://10.0.0.1:88/MyCollection/movie8126.dcf
</res>

<res protocolInfo=
 "http-get:*:application/x-dtcp1;CONTENTFORMAT=video/MP2T:*"
 allowedUse="PLAY,5"
 validityStart="2004-05-30T14:30:00"
 validityEnd="2004-06-04T14:30:00">
   http://10.0.0.1:88/MyCollection/movie9736.mp2t?
   CONTENTPROTECTIONTYPE=DTCP1&amp;DTCP1HOST=1.2.3.4&amp;DTCP1PORT
=97</res>
```

2.5.2.4.2 4th Field - Convention for Protected Content

The UPnP AV WC additionally defines the following convention for placing information in the fourth field of the *ProtocolInfo* string. The fourth field is used to convey additional information in cases when MIME type is not sufficient for the purpose of capability matching. In these cases the fourth field MUST contain additional DRM information for the purpose of more precise compatibility checking between the media sink and the content properties. Two cases are possible. In the first case, it is only required to identify the vendor or the standards group that defines the DRM scheme. In this case, the DRM organization or vendor is specified as the value of a upnp-domain variable:

upnp.org_DRMInfo=<ICANN-DRM-ORG-NAME>

In the second case, the DRM scheme also requires one or more parameters associated with the DRM scheme to be enumerated. In this case, the following convention is used:

upnp.org_DRMInfo=<ICANN-DRM-ORG-NAME>;
<ICANN-DRM-ORG-NAME>_<parameter1>=<parameter1 value>;
<ICANN-DRM-ORG-NAME>_<parameter2>=<parameter2 value>;
…
<ICANN-DRM-ORG-NAME>_<parameterN>=<parameterN value>

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Example 4: A *ProtocolInfo* string utilizes the fourth field to indicate that the MPEG-4 content is protected by a (fictitious) DRM scheme associated with ICANN name "XYZ.ORG":

ProtocolInfo = "http-get:*:video/mp4:upnp.org_DRMInfo=XYZ.ORG"

Example 5: A *ProtocolInfo* string for an OMA dcf file also indicates the OMA version in the fourth field:

ProtocolInfo = "http-get:*:application/vnd.oma.drm.dcf;CONTENTFORMAT=video/MP2P:
upnp.org_DRMInfo=OMA.ORG;OMA.ORG_VERSION=2"

For maximal compatibility checking, both third and fourth fields (when present) of the *ProtocolInfo* string should be matched. In general, older control points may not be capable of checking the fourth field of the *ProtocolInfo* string. For that reason, it is recommended that content listings in the CDS for DRM content should use the MIME conventions described above as much as possible for inserting DRM information on the given content item into the third field of the *ProtocolInfo* string.

2.5.3 Typical Control Point Operations

This section briefly outlines some typical control point operations on a ConnectionManager service.

2.5.3.1 Establishing a New Connection

The process for establishing a streaming connection involves:

1. Find ConnectionManager services via SSDP
2. Determine compatibility between the source (sending) and the sink (receiving) device (See Section 2.4.1)
3. Invoke *PrepareForConnection()* on the source and/or sink devices, if the action is implemented by the device (See Section 2.4.2)
4. Transfer content from source to sink device. (See note below).
5. When the connection is no longer needed, invoke *ConnectionComplete()* on the source and/or sink devices, if the action is implemented by the device. (See Section 2.4.3)

Refer to the UPnP AV Architecture document for additional details.

Once a connection has been prepared, it can be used to transfer several pieces of content before calling *ConnectionComplete()* as long as each content item is compatible with the *RemoteProtocolInfo* argument that was passed into *PrepareForConnection()*; that is: each content item has the same media format as specified in *RemoteProtocolInfo*.

2.5.3.2 Dealing with Ongoing Connections

A number of interesting scenarios require a control point to find information about all currently ongoing connections in the network, including those that it did not establish itself. This is supported by the ConnectionManager as follows. Each connection explicitly established by any control point in the network is identified by a connection identifier on both the source (sending) device and the sink (receiving) device. State variable *CurrentConnectionIDs* holds a Comma-Separated Value list of these identifiers. Given an identifier, a control point can call *GetConnectionInfo()* to obtain:

- The *ProtocolInfo* of the connection. This includes the streaming protocol and the content format.
- The 'other end' of the connection, expressed as a **UDN**/**serviceId** pair. Using the **UDN**, a control point can use SSDP to find the device description of the other UPnP device involved in the connection. This way, a control point can find out, for example, that turning off a particular source device is going to affect one or more sink devices.
- The connection status.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- The *AVTransportID* of the connection, which indicates the AVTransport service instance controlling the playback and recording through the connection. This service can be used for many purposes, for example to:
    - subscribe to events in order to monitor the transport state
    - actually change the transport state, for example, stopping or pausing an existing stream
    - obtain a URI reference to the content resource currently flowing through the connection
    - obtain any meta data embedded in the content resource flowing through the connection.

  See the AVTransport service description for more details.
- The *RcsID* of the connection, which indicates the RenderingControl service instance controlling the rendering properties of the content. This can be used, for example, to implement a 'mute all streams' function in a control point.

2.5.4 Relation to Devices without ConnectionManagers

In some cases, it is desirable to establish a stream connection between devices where one device implements a UPnP ConnectionManager service, and the other device doesn't implement this service or isn't even a UPnP device. In such cases, a control point can only call *PrepareForConnection()* and *ConnectionComplete()* actions on the first device. The *PeerConnectionManager* input argument to *PrepareForConnection()* is defined as the **UDN** of the connecting UPnP device followed by a slash ('/') and the **serviceId** of the connecting device's ConnectionManager service. In case the connecting UPnP device has no ConnectionManager service, the **serviceId** part of the argument is left blank. In case the connecting device is not a UPnP device (for example, an Internet streaming server), the whole PeerConnectionManager argument is left blank.

2.5.5 *PrepareForConnection()* and *ConnectionComplete()*

2.5.5.1 *PrepareForConnection()*

The purpose of *PrepareForConnection()* is to allow a device to perform a set of tasks prior to transferring the content. The specific tasks performed by a device are implementation dependent, but may include the following:

- Determine whether or not the device is able to stream content using the current environment (for example, device status, network conditions, etc.)
- Allocating some resources that are needed to establish the out-of-band connection between the source/sink devices for example, in an IEEE-1394/IEC-61883 environment, this may include allocating an IEEE-1394 isochronous channel.
- Allocating a unique *ConnectionID* that identifies those resources which were allocated for a specific connection.
- Allocating a new virtual instance of the AVTransport and/or RenderingControl service and binding it to the connection so that the flow of the content and the rendering characteristics of the content can be controlled.

If a control point wants to interoperate with all UPnP AV devices, prior to initiating the transfer of content, for example, invoking *AVTransport::SetAVTransportURI()*, the control point needs to invoke *PrepareForConnection()*, if the action is implemented by the device. Otherwise, the device may not operate correctly because it is not yet properly configured. Additionally, the control point will not know whether or not the current environment is able to support the upcoming streaming request.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.5.5.2 *ConnectionComplete()*

The purpose of *ConnectionComplete()* is to allow a device to terminate a specific connection and/or to perform any cleanup tasks that are needed for the connection as a result of a previous invocation of *PrepareForConnection()*. As with *PrepareForConnection()*, the set of tasks performed by a device when *ConnectionComplete()* is invoked is implementation-dependent, but may include the following:

- Releasing the resources that were allocated to establish the out-of-band connection between the source and sink devices (for example, in a IEEE-1394/IEC-61883 environment, this may include releasing the IEC-61883 isochronous channel that was allocated when *PrepareForConnection()* was invoked earlier on the same device).
- Releasing the unique *ConnectionID* that identifies those resources that were allocated by a previous invocation of *PrepareForConnection()*.
- Releasing the virtual instance of the AVTransport and/or RenderingControl service, if any, that were allocated during a previous invocation of *PrepareForConnection()* in order to control the content flowing over the associated connection.

Since control points may turn off after a connection is established, control points may not always invoke the *ConnectionComplete()* action. Therefore, the device needs to automatically perform any cleanup tasks for the connection so that those resources that were allocated during *PrepareForConnection()* are not *leaked*.

2.5.5.3 General Usage Model

As mentioned earlier, each device performs an arbitrary set of implementation-dependent tasks during *PrepareForConnection()* and *ConnectionComplete()*. Some of these tasks may be crucial to the proper operation of the device while other tasks may be secondary to the device's core functionality. However, since each implementation of *PrepareForConnection()* and *ConnectionComplete()* are specific to each device, it is very difficult (if not impossible) for a control point to determine whether or not it is safe to by-pass *PrepareForConnection()*/*ConnectionComplete()* for a given device. Therefore, the safest and simplest way for a control point to interoperate with all UPnP AV devices is to always invoke *PrepareForConnection()* and *ConnectionComplete()* if they are implemented by the device. Otherwise, those devices may not function properly as described above.

2.5.5.4 Relationship to AVTransport and RenderingControl Services

As described in the "Theory of Operation" sections of the AVTransport and RenderingControl service specifications, some device are designed to support multiple virtual instances of the AVTransport and/or RenderingControl service. With these types of devices, the allocation and binding of these virtual instances occur during *PrepareForConnection()*.

As described in the AVTransport specification, the responsibility for providing the AVTransport service for a given connection varies between the source and sink devices depending on the type of connection (that is: the type of transfer protocol that is being used). When a *push* protocol is being used (for example, IEEE-1394), the source device is responsible for providing the AVTransport service and when a *pull* protocol is being used (for example, HTTP GET), the sink device is responsible for providing the AVTransport service. When a source device wants to support multiple instances of a *push* protocol or a sink device wants to support multiple instances of a *pull* protocol, the device's *PrepareForConnection()* is responsible for allocating a new virtual instance of the AVTransport service for each new instance of that connection-type. Additionally, *PrepareForConnection()* MUST perform the necessary binding operations that link the allocated AVTransport instance with the connection.

For example, in a IEEE-1394/IEC-61883 (*push*) environment, if a source device wants to support multiple 1394/61883 streams, then its *PrepareForConnection()* MUST allocate a unique virtual instance of the AVTransport service and bind it to the newly allocated IEEE-1394/IEC-61883 connection. Similarly, in an HTTP GET *pull* environment, if a sink device wants to support multiple simultaneous connections, its *PrepareForConnection()* implementation MUST allocate a new virtual instance of the AVTransport service and bind it to the newly allocated connection.

Note: This implies that the sink device's *PrepareForConnection()* MUST perform some type of pre-allocation of the TCP/IP socket so that it can be distinguished from the other connections of that type.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

With regards to the RenderingControl service, the sink device is always responsible for providing it regardless of the underlying protocol. If a sink device is designed to support multiple simultaneous connections then its implementation of *PrepareForConnection()* MUST be designed to allocate a new virtual instance of the RenderingControl service for each connection that is created. Additionally, it MUST bind each instance to the newly created connection so that a control point can control the rendering characteristics of the content that is being transferred over that connection.

Note: This implies that the sink device's *PrepareForConnection()* MUST perform some type of pre-allocation of the TCP/IP socket so that it can be distinguished from the other connections of that type.

When a device's *PrepareForConnection()* is designed to allocate and bind virtual instances of the AVTransport and/or RenderingControl services, the device's *ConnectionComplete()* MUST be designed to un-bind and release these virtual instances. For example, if a source device allocates an AVTransport service and binds it to a IEEE-1394 channel, the device's *ConnectionComplete()* action MUST undo the binding operation, as appropriate, and it MUST release the IEEE-1394 channel.

2.5.5.5 *ConnectionIDs*

A *ConnectionID* is a device-specific identifier that is used to uniquely identify a connection which has been prepared on a device via *PrepareForConnection()*. When *PrepareForConnection()* is invoked, a *ConnectionID* is allocated by the device and assigned to the newly configured connection. Since *ConnectionID*s are allocated by individual devices, a given *ConnectionID* is valid only within the context of that device. Therefore, a *ConnectionID* assigned by one device can not be used when interacting with another device. When the two end-point devices of a given connection are setup via *PrepareForConnection()*, the *ConnectionID*s returned by the two devices are completely independent from each other and are almost certainly going to have different values even though they happen to refer to the same connection.

On a given device, the algorithm used to allocate *ConnectionID*s is vendor-specific. Hence, the numerical value of a *ConnectionID* is completely meaningless except to the device itself. Once a *ConnectionID* has been allocated, it is generally valid until the associated connection is torn down. Typically, this happens in response to an invocation of *ConnectionComplete()* or as a result of the device's OPTIONAL auto-cleanup mechanism.

Devices SHOULD NOT return the same *ConnectionID* value on subsequent invocations of *PrepareForConnection()*. After a connection has been torn down, its associated *ConnectionID*, which is no longer valid, can be reassigned by the device to another connection. However, in order to minimize the potential of a *stale* *ConnectionID* being misinterpreted as a *valid* *ConnectionID*, it is RECOMMENDED that each device not reassign a *ConnectionID* value until all other valid values have been used.

Once a *ConnectionID* has been allocated, any control point may use the *ConnectionID* to uniquely identify the associated connection even when invoking *ConnectionComplete()*. However, in order to provide predictable device behavior, it is RECOMMENDED that each control point use only those connections that it has prepared. Notable exceptions to this recommendation include those control points that are able to coordinate with one another, via some non-UPnP mechanism, or those control points that are explicitly designed to perform connection clean up tasks, for example, a network management tool.

2.5.5.6 AVTransportIDs and RcsIDs

When a connection is prepared via *PrepareForConnection()*, the device MAY choose to return an *AVTransportID* and/or an *RcsID*. These IDs are used to identify the (unique and independent) virtual instance of the AVTransport service and/or RenderingControl service that has been associated with the newly prepared connection. As with the connection's *ConnectionID*, the value of the *AVTransportID* and/or *RcsID* have no meaning outside of the context of the allocating device. Similarly, *AVTransportID*s and *RcsID*s are valid until their associated connection is torn down, generally in response to an invocation of *ConnectionComplete()* or as a result of the device's OPTIONAL auto-cleanup mechanism.

Once allocated, *AVTransportID*s and *RcsID*s are used in conjunction with the AVTransport service and RenderingControl service, respectively, to invoke various control actions on the stream that is being

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

carried over the associated connection. As with *ConnectionID*s, control points should use only the *AVTransportID*s and *RcsID*s that are associated with the connections that the control point has prepared via *PrepareForConnection()*. The AVTransport and RenderingControl services allow any control point to use any valid *AVTransportID*s and/or *RcsID*s. However, when multiple control points use the same *AVTransportID* and/or *RcsID*, these control points should coordinate their activities with one another. Otherwise, the devices may behave unexpectedly thus causing a poor end-user experience.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3 XML Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>
   <actionList>
      <action>
         <name>GetProtocolInfo</name>
         <argumentList>
            <argument>
               <name>Source</name>
               <direction>out</direction>
               <relatedStateVariable>
                  SourceProtocolInfo
               </relatedStateVariable>
            </argument>
            <argument>
               <name>Sink</name>
               <direction>out</direction>
               <relatedStateVariable>
                  SinkProtocolInfo
               </relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>PrepareForConnection</name>
         <argumentList>
            <argument>
               <name>RemoteProtocolInfo</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ProtocolInfo
               </relatedStateVariable>
            </argument>
            <argument>
               <name>PeerConnectionManager</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ConnectionManager
               </relatedStateVariable>
            </argument>
            <argument>
               <name>PeerConnectionID</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_ConnectionID
               </relatedStateVariable>
            </argument>
            <argument>
               <name>Direction</name>
               <direction>in</direction>
               <relatedStateVariable>
                  A_ARG_TYPE_Direction
               </relatedStateVariable>
            </argument>
            <argument>
               <name>ConnectionID</name>
               <direction>out</direction>
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
                <relatedStateVariable>
                   A_ARG_TYPE_ConnectionID
                </relatedStateVariable>
             </argument>
             <argument>
                <name>AVTransportID</name>
                <direction>out</direction>
                <relatedStateVariable>
                   A_ARG_TYPE_AVTransportID
                </relatedStateVariable>
             </argument>
             <argument>
                <name>RcsID</name>
                <direction>out</direction>
                <relatedStateVariable>
                   A_ARG_TYPE_RcsID
                </relatedStateVariable>
             </argument>
          </argumentList>
       </action>
       <action>
          <name>ConnectionComplete</name>
          <argumentList>
             <argument>
                <name>ConnectionID</name>
                <direction>in</direction>
                <relatedStateVariable>
                   A_ARG_TYPE_ConnectionID
                </relatedStateVariable>
             </argument>
          </argumentList>
       </action>
       <action>
          <name>GetCurrentConnectionIDs</name>
          <argumentList>
             <argument>
                <name>ConnectionIDs</name>
                <direction>out</direction>
                <relatedStateVariable>
                   CurrentConnectionIDs
                </relatedStateVariable>
             </argument>
          </argumentList>
       </action>
       <action>
          <name>GetCurrentConnectionInfo</name>
          <argumentList>
             <argument>
                <name>ConnectionID</name>
                <direction>in</direction>
                <relatedStateVariable>
                   A_ARG_TYPE_ConnectionID
                </relatedStateVariable>
             </argument>
             <argument>
                <name>RcsID</name>
                <direction>out</direction>
                <relatedStateVariable>
                   A_ARG_TYPE_RcsID
                </relatedStateVariable>
             </argument>
             <argument>
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
            <name>AVTransportID</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_AVTransportID
            </relatedStateVariable>
         </argument>
         <argument>
            <name>ProtocolInfo</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_ProtocolInfo
            </relatedStateVariable>
         </argument>
         <argument>
            <name>PeerConnectionManager</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_ConnectionManager
            </relatedStateVariable>
         </argument>
         <argument>
            <name>PeerConnectionID</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_ConnectionID
            </relatedStateVariable>
         </argument>
         <argument>
            <name>Direction</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_Direction
            </relatedStateVariable>
         </argument>
         <argument>
            <name>Status</name>
            <direction>out</direction>
            <relatedStateVariable>
               A_ARG_TYPE_ConnectionStatus
            </relatedStateVariable>
         </argument>
      </argumentList>
   </action>
</actionList>
<serviceStateTable>
   <stateVariable sendEvents="yes">
      <name>SourceProtocolInfo</name>
      <dataType>string</dataType>
   </stateVariable>
   <stateVariable sendEvents="yes">
      <name>SinkProtocolInfo</name>
      <dataType>string</dataType>
   </stateVariable>
   <stateVariable sendEvents="yes">
      <name>CurrentConnectionIDs</name>
      <dataType>string</dataType>
   </stateVariable>
   <stateVariable sendEvents="no">
      <name>A_ARG_TYPE_ConnectionStatus</name>
      <dataType>string</dataType>
      <allowedValueList>
         <allowedValue>OK</allowedValue>
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                <allowedValue>ContentFormatMismatch</allowedValue>
                <allowedValue>InsufficientBandwidth</allowedValue>
                <allowedValue>UnreliableChannel</allowedValue>
                <allowedValue>Unknown</allowedValue>
            </allowedValueList>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_ConnectionManager</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_Direction</name>
            <dataType>string</dataType>
            <allowedValueList>
                <allowedValue>Input</allowedValue>
                <allowedValue>Output</allowedValue>
            </allowedValueList>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_ProtocolInfo</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_ConnectionID</name>
            <dataType>i4</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_AVTransportID</name>
            <dataType>i4</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>A_ARG_TYPE_RcsID</name>
            <dataType>i4</dataType>
        </stateVariable>
    </serviceStateTable>
</scpd>
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4 Test

No semantics tests have been defined for this service.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Annex A
(normative)

Protocol Specifics

A.1 Application to HTTP Streaming

A.1.1 *ProtocolInfo* Definition

Streaming data via the HTTP GET method is defined by the Internet standard Request For Comment document entitled HyperText Transport Protocol – HTTP/1.1 (http://www.ietf.org/rfc/rfc2616.txt). While it is certainly possible to use other HTTP methods such as PUT or POST, this document focuses on the HTTP GET method. The <protocol> part of *ProtocolInfo* MUST be "`http-get`". The <network> part of *ProtocolInfo* is not used for the HTTP case since all devices belong to the same network. An asterisk ("*") is used instead. The <contentFormat> part for HTTP GET is described by a MIME type, see http://www.ietf.org/rfc/rfc1341.txt.

An example of protocol information for HTTP GET, in this case referring to an audio file, is:

```
http-get:*:audio/mpeg:*
```

A.1.2 Implementation of *PrepareForConnection()*

In addition to any non-protocol related preparation tasks such as the one described in Section 2.5.5, "*PrepareForConnection()* and *ConnectionComplete()*," a device's *PrepareForConnection()* implementation MAY also perform some preparation tasks that are related to the protocol that is about to be used to transfer the content. However, since the HTTP GET connection is initiated and maintained by the sink device, the source device typically does not need to perform any protocol-related preparation tasks because HTTP GET requests are handled by the device's underlying http-server. Therefore, if a MediaServer does not need to perform any non-protocol-related preparation tasks, it will (in most cases) not need to implement *PrepareForConnection()*. Although not required, the MediaRenderer device (the receiving end of the HTTP stream), MAY choose to pre-allocate a TCP/IP socket in order to ensure that this resource is available when the content transfer is initiated; that is: when *AVTransport::SetAVTransportURI()* is invoked.

A.1.3 Implementation of *ConnectionComplete()*

In addition to the non-protocol related cleanup tasks such as those described in Section 2.5.5, "*PrepareForConnection()* and *ConnectionComplete()*," a device's *ConnectionComplete()* implementation MAY also perform some cleanup tasks that are related to the protocol that was used to transfer the content. The cleanup tasks that a device performs depend directly on the implementation of *PrepareForConnection()*. In general, when using the HTTP GET protocol, a MediaServer does not have any protocol-related cleanup tasks to perform because the MediaServer's *PrepareForConnection()* typically does not perform any protocol-related preparation. On a MediaRenderer device, *ConnectionComplete()* MUST release any protocol-related resources that were allocated during *PrepareForConnection()*. For example, if a MediaRenderer chooses to pre-allocate a TCP/IP socket during *PrepareForConnection()*, the device's *ConnectionComplete()* action MUST release the socket associated with that connection.

A.1.4 Automatic Connection Cleanup

Since control points may establish connections, and then leave the UPnP network forever, protocols supported by the ConnectionManager need to have a built-in automatic mechanism to cleanup stale connections. For HTTP connections, automatic cleanup SHOULD be performed by the AVTransport instance.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

On the UPnP level, this will appear as an (evented) change in state variable *CurrentConnectionIDs*.

A.2 Application to RTSP/RTP/UDP Streaming

A.2.1 *ProtocolInfo* Definition

Streaming data via RTSP is defined by the Internet standard Request For Comment document entitled Real Time Streaming Protocol. (http://www.ietf.org/rfc/rfc2326.txt). The actual Audio/Video data packets are sent out-of-band with respect to RTSP. RTSP does not REQUIRE a particular protocol for this. Since usually RTP (http://www.ietf.org/rfc/rfc1889.txt) over UDP is used, the protocol for RTSP-based streams is defined as RTSP/RTP/UDP. This ensures that two ConnectionManagers that can send and receive RTSP also send and receive using the same Audio/Video data Connection protocol. The <protocol> part of *ProtocolInfo* MUST be "`rtsp-rtp-udp`". The <network> part of *ProtocolInfo* is not used for the RTSP/RTP/UDP case since all devices belong to the same network. An asterisk ("*") is used instead. RTP packets contain a standardized 7-bit payload type identifier, see http://www.iana.org/assignments/rtp-parameters or http://www.ietf.org/rfc/rfc1890.txt. Each payload type has a unique encoding name. This payload type name is used as the <contentFormat> part of the *ProtocolInfo* string.

An example of protocol information for RTSP/RTP/UDP with MPEG video payload is:

```
rtsp-rtp-udp:*:MPV:*
```

A.2.2 Implementation of *PrepareForConnection()*

In addition to the non-protocol related preparation task such as those described in Section 2.5.5, "*PrepareForConnection()* and *ConnectionComplete()*," a device's *PrepareForConnection()* implementation MAY also perform some preparation tasks that are related to the protocol that is about to be used to transfer the content. However, since RTSP/RTP/UDP sessions are initiated and maintained by the sink device, the source device typically does not need to perform any protocol-related preparation tasks. Therefore, if a MediaServer does not need to perform any non-protocol-related preparation tasks, it will (in most cases) not need to implement *PrepareForConnection()*. Although not required, the MediaRenderer device (the receiving end of the RTP/RTP/UDP stream) MAY choose to pre-allocate an RTSP/RTP/UDP connection in order to ensure that this resource is available when the content transfer is initiated; that is: when *AVTransport::SetAVTransportURI()* is invoked.

A.2.3 Implementation of *ConnectionComplete()*

In addition to the non-protocol related cleanup tasks such as those described in Section 2.5.5, "*PrepareForConnection()* and *ConnectionComplete()*," a device's *ConnectionComplete()* implementation MAY also perform some cleanup tasks that are related to the protocol that was used to transfer the content. The cleanup tasks that a device performs depend directly on the implementation of *PrepareForConnection()*. In general, when using the RTSP/RTP/UDP protocol, a MediaServer does not have any protocol-related cleanup tasks to perform because the MediaServer's *PrepareForConnection()* typically does not perform any protocol-related preparation. On a MediaRenderer device, *ConnectionComplete()* MUST release any protocol-related resources that were allocated during *PrepareForConnection()*. For example, if a MediaRenderer chooses to pre-allocate a RTSP/RTP/UDP connection during *PrepareForConnection()*, the device's *ConnectionComplete()* MUST release that connection.

A.2.4 Automatic Connection Cleanup

Since control points may establish connections, and then leave the UPnP network forever, protocols supported by the ConnectionManager need to have a built-in automatic mechanism to cleanup stale connections. For RTSP connections, automatic cleanup SHOULD be performed by the AVTransport instance.

On the UPnP level, this will appear as an (evented) change in state variable *CurrentConnectionIDs*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

A.3 Application to Device-Internal Streaming

For the purpose of this service definition an INTERNAL protocol is defined for use over internal connections. An internal connection is defined as a connection within a single device. An example of such a connection is between a Tuner subsystem and a Display subsystem in a conventional TV. Since this connection is internal to the device, no streaming data will flow on the UPnP network, and the actual content-format used inside the device can be proprietary. The resulting *ProtocolInfo* and content-URI that need to be defined for these types of connections can therefore be very simple.

An internal connection MUST use the INTERNAL protocol. For this protocol, the <protocol> part of *ProtocolInfo* MUST be "*internal*". Within this protocol scope the <network> part of *ProtocolInfo* is defined as the device's IP-address, as a string, in the well-known dotted decimal notation. The <contentFormat> part of *ProtocolInfo* is proprietary.

An example of protocol information for INTERNAL is:

```
internal:161.88.59.212:mpeg2:to-local-display
```

The implementation of the *PrepareForConnection()* and *ConnectionComplete()* actions for this protocol type is proprietary (vendor specific).

A.4 Application to IEC61883 Streaming

A.4.1 *ProtocolInfo* Definition

The basis for real time data transmission on the IEEE1394 bus using the IEC61883 protocol is the Common Isochronous Packet (CIP) which consists of a CIP header and data blocks embedded in an IEEE1394 compliant isochronous packet. The <protocol> part of *ProtocolInfo* MUST be "`iec61883`".

The <network> part of *ProtocolInfo* for the IEC61883 protocol uniquely identifies the set of connected IEEE1394 devices on a specific bus segment. It is defined as a bin.hex encoding of the GUID (Globally Unique ID) of the 1394 Isochronous Resource Manager node. This identification is not persistent, and will, in general, change when 1394 devices are added to or removed from the 1394 network. These changes will lead to changes in the *SourceProtocolInfo* and *SinkProtocolInfo* state variables, and, through eventing, interested control points will be notified of the new streaming possibilities of the new 1394 network segmentation.

The stream types include all content formats supported by the family of IEC61883 Standards. These formats are uniquely identified by the FMT and FDF values in the CIP header, The following table lists the formats supported by the IEC61883-2 to 5 International Standards and by IEC61883-6 PAS (Publicly Available Specification; *that is: not yet fulfilling all requirements for a standard*).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table A-1: <contentFormat> for Protocol IEC61883

<contentFormat> for Protocol IEC61883	IEC Version	Description
"*UNKNOWN_STREAM*"		
"*DVCR_STD_DEF_525_60*"	IEC61883-2	525-60 system: the 525-line system with a frame frequency of 29.97 Hz
"*DVCR_STD_DEF_625_50*"	IEC61883-2	625-50 system: the 625-line system with a frame frequency of 25.00 Hz
"*DVCR_STD_DEF_HI_COMPRESS_525_60*"	IEC61883-5	SDL525-60 system: The standard definition for high compression mode 525-line system with a frame frequency of 29.97 Hz.
"*DVCR_STD_DEF_HI_COMPRESS_625_50*"	IEC61883-5	SDL625-50 system: The standard definition for high compression mode 625-line system with a frame frequency of 25.00 Hz.
"*DVCR_HI_DEF_1125_60*"	IEC61883-3	1125-60 system: the 1125-line system with a frame frequency of 30.00 Hz
"*DVCR_HI_DEF_1250_50*"	IEC61883-3	1250-50 system: the 1250-line system with a frame frequency of 25.00 Hz
"*SMPTE_D7_525_60*"	IEC61883-2	SMPTE-D7 525-60 system
"*SMPTE_D7_625_50*"	IEC61883-2	SMPTE-D7 625-50 system
"*MPEG2_TS*"	IEC61883-4	MPEG2-TS
"*AUDIO_MUSIC_8_24_IEC_60958*"	IEC61883-6	IEC 60958 conformant
"*AUDIO_MUSIC_8_24_RAW_AUDIO*"	IEC61883-6	Raw audio
"*AUDIO_MUSIC_8_24_MIDI*"	IEC61883-6	MIDI conformant

IEC61883 connections are set up between iPCRs (input Plug Control Registers) and oPCRs (output Plug Control Registers). A content item is connected through an oPCR to one or more iPCRs on a different device. An IEC61883 device can have zero or more iPCRs and oPCRs.

The <additionalInfo> field identifies the PCR in the IEC61883 network, and is defined as follows:

```
<GUID>;<PCR-index>
```

where

- <GUID> = bin.hex encoding of the device's node_vendor_id and chip_id (2 quadlets, together also referred to as GUID)
- <PCR-index> = zero-based integer index identifying the plug within the device

An example of protocol information for IEC61883 is:

```
iec61883:0000f00200001114:MPEG2_TS:00ba0091c9231222;0
```

A.4.2 Implementation of *PrepareForConnection()*

In addition to the non-protocol related preparation task such as those described in Section 2.5.5, "*PrepareForConnection()* and *ConnectionComplete()*," a device's *PrepareForConnection()* implementation MAY also perform some preparation tasks that are related to the protocol that is about to be used to transfer the content Although IEEE1394/IEC61883 is an allocation-based protocol, the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

source device is not REQUIRED to perform any protocol related preparation. It is the sink device that is responsible for allocating the underlying IEEE1394/IEC61883 connection.

In order to manage isochronous data transmissions, IEC61883 defines the concept of plug and specialized registers called MPR (Master Plug Register) and PCR (Plug Control Register). These registers are used to initiate and stop transmissions. The set of procedures to control the real time data flow by manipulating the PCRs is called CMP (Connection Management Procedures). Data transmission between devices is possible when an output plug on the source device is connected to an input plug on the sink device via an isochronous channel. The data flow from a source device is controlled by the oMPR (output Master Plug Register) of the device and one oPCR (output PCR). Similarly, the data flow to a sink device is controlled by the iMPR (input MPR) and one iPCR (input PCR). The address map for these registers is well defined in conformance with ISO/IEC13213 (ANSI/IEEE1212). Devices can modify PCR values of remote nodes using asynchronous transactions.

Using the information in the *PrepareForConnection()* input arguments, the sink device locates the IEEE1394 address of the source device (its GUID is part of the <additionalInfo> field of the *ProtocolInfo* string), and program the appropriate oPCR register to initiate the streaming. The sink device is free to choose any of its own iPCRs. The sink device MUST follow the exact procedure defined by IEC61883, which includes the allocation of IEEE1394 bandwidth and an IEEE1394 channel. Upon subsequent IEEE1394 bus resets, the *sink* device (the device that established the connection) MUST try to restore any existing connections that it has established.

If the *ProtocolInfo* references an oPCR that is already in use, two situations occur:

- The same content-format is already being streamed via the oPCR. In this case, the sink device performs an IEC61883 *overlay* connection.
- A different content-format is already being streamed via the oPCR. In this case, the sink device will return an error.

IEC61883 broadcast-in and broadcast-out connections are not supported by the ConnectionManager.

A.4.3 Implementation of *ConnectionComplete()*

In addition to the non-protocol related cleanup tasks such as those described in Section 2.5.5, "*PrepareForConnection()* and *ConnectionComplete()*," a device's *ConnectionComplete()* implementation MAY also perform some cleanup tasks that are related to the protocol that was used to transfer the content. The cleanup tasks that a device performs depend directly on the implementation of *PrepareForConnection()*. In general, when using the IEEE1394/IEC61883 protocol, the source device does not have any protocol-related cleanup tasks to perform because the device's *PrepareForConnection()* typically does not perform any protocol-related preparation. On a sink device, *ConnectionComplete()* MUST release the IEEE1394/IEC61883 connection that was allocated during *PrepareForConnection()*. The sink device MUST follow the IEC61883 procedure for releasing the channel which includes:

- Modifying the corresponding fields of the source oPCR and sink iPCR according to CMP procedures.
- Deallocate the IEEE1394 resources. If the oPCR becomes unconnected (that is: this is the last IEC61883 connection on that IEEE1394 channel), the IEEE1394 bandwidth and channel MUST also be released.

Note: IEC61883 broadcast-in and broadcast-out connections are not supported by the ConnectionManager.

A.4.4 Automatic Connection Cleanup

Since control points may establish connections, and then leave the UPnP network forever, protocols supported by the ConnectionManager need to have a built-in automatic mechanism to cleanup stale connections. For the IEC61883 protocol, an established connection will continue forever, until there is a so-called *bus reset*. A bus reset will occur when there is a change in the physical network topology, for example, the network is split, joined with another network, or a device goes offline. After a bus reset, all IEEE1394 resources are released, and all devices that established IEC61883 connections have

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

1 second to re-establish them. Hence, the ConnectionManager on the sink device needs to check after a bus reset whether the source device is still on the network, and if not, cleanup any internal state referring to this connection. On the UPnP level, this will appear as an (evented) change in state variable *CurrentConnectionIDs*.

A.5 Application to Vendor-specific Streaming

To allow vendors to use their vendor-specific streaming protocols in a UPnP network in a controlled way, the ConnectionManager defines the generic protocol VENDOR for such protocols. The idea is to make the <protocol> part of *ProtocolInfo* unique, by requiring the use of the vendor's registered ICANN (Internet) domain name (similar to its use in vendor-specific UPnP service- and device-types). The remaining fields of the *ProtocolInfo* string (<network>, <contentFormat> and <additionalInfo>) are all vendor-specific, and MAY be wildcards ("*").

An example of a VENDOR protocol information is:

```
company.com:*:company-format-A:optional-setup-info
```

The implementation of the *PrepareForConnection()* and *ConnectionComplete()* actions for this protocol type is proprietary (vendor-specific).

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

Tel: + 41 22 919 02 11
Fax: + 41 22 919 03 00
info@iec.ch
www.iec.ch

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale