

OpenCable Specifications Home Networking 2.0

Home Networking Protocol 2.0

OC-SP-HNP2.0-I03-100603

ISSUED

Notice

This OpenCable specification is the result of a cooperative effort undertaken at the direction of Cable Television Laboratories, Inc. for the benefit of the cable industry and its customers. This document may contain references to other documents not owned or controlled by CableLabs. Use and understanding of this document may require access to such other documents. Designing, manufacturing, distributing, using, selling, or servicing products, or providing services, based on this document may require intellectual property licenses from third parties for technology referenced in this document.

Neither CableLabs nor any member company is responsible to any party for any liability of any nature whatsoever resulting from or arising out of use or reliance upon this document, or any document referenced herein. This document is furnished on an "AS IS" basis and neither CableLabs nor its members provides any representation or warranty, express or implied, regarding the accuracy, completeness, noninfringement, or fitness for a particular purpose of this document, or any document referenced herein.

© Copyright 2007-2010 Cable Television Laboratories, Inc.
All rights reserved.

Document Status Sheet

Document Control Number:	OC-SP-HNP2.0-I03-100603			
Document Title:	Home Networking Protocol 2.0			
Revision History:	I01 - Released 4/18/08			
	I02 - Released 12/17/09			
	I03 - Released 6/3/10			
Date:	June 3, 2010			
Status:	Work in Progress	Draft	Issued	Closed
Distribution Restrictions:	Author Only	CL/Member	CL/Member/ Vendor	Public

Key to Document Status Codes:

- Work in Progress** An incomplete document, designed to guide discussion and generate feedback, that may include several alternative requirements for consideration.
- Draft** A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process.
- Issued** A stable document, which has undergone rigorous member and vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing.
- Closed** A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs.

Trademarks

CableLabs[®], DOCSIS[®], EuroDOCSIS[™], eDOCSIS[™], M-CMTS[™], PacketCable[™], EuroPacketCable[™], PCMM[™], CableHome[®], CableOffice[™], OpenCable[™], OCAP[™], CableCARD[™], M-Card[™], DCAS[™], tru2way[™], and CablePC[™] are trademarks of Cable Television Laboratories, Inc.

Contents

1	SCOPE	1
1.1	Introduction and Overview.....	1
1.2	Purpose of document.....	1
1.3	Requirements.....	1
2	REFERENCES	2
2.1	Normative References.....	2
2.2	Informative References.....	3
2.3	Reference Acquisition.....	4
3	TERMS AND DEFINITIONS	5
4	ABBREVIATIONS AND ACRONYMS	6
5	TECHNICAL REQUIREMENTS	7
5.1	UPnP Protocol.....	7
5.2	DLNA Compliance.....	8
5.3	Networking and Connectivity.....	9
5.4	Device Discovery and Control.....	9
5.4.1	Eventing.....	9
5.5	Media Management.....	10
5.5.1	Device Namespace.....	10
5.5.2	Metadata Namespace.....	11
5.5.3	Middleware Elements.....	11
5.6	Media Transport.....	11
5.6.1	HTTP Headers.....	12
5.7	Content Transformation Device Virtualization.....	17
5.8	Content Formats.....	17
5.9	Quality of Service (QoS).....	18
5.10	Content Protection.....	18
5.11	UPnP Interoperability.....	18
6	UPNP MAPPING REQUIREMENTS	20
6.1	Device Registry.....	20
6.2	Package org.ocap.hn.....	20
6.2.1	ContentServerNetModule interface.....	20
6.2.2	Device interface.....	22
6.2.3	DeviceEvent class.....	23
6.2.4	NetActionEvent class.....	24
6.2.5	NetActionRequest.....	24
6.2.6	HomeNetPermission class.....	25
6.2.7	NetManager class.....	25
6.2.8	NetModuleEvent class.....	25
6.2.9	NetModule interface.....	25
6.2.10	PropertyFilter class.....	26
6.3	Package org.ocap.hn.content.....	26
6.3.1	AudioResource Interface.....	26
6.3.2	ContentContainer.....	26
6.3.3	ContentEntry Interface.....	29
6.3.4	ContentResource class.....	30
6.3.5	IOStatus interface.....	31

6.3.6	<i>MetadataNode class</i>	31
6.3.7	<i>VideoResource Interface</i>	34
6.3.8	<i>StreamableContentResource interface</i>	34
6.4	Package <i>Org.ocap.hn.content.navigation</i>	34
6.4.1	<i>ContentDatabaseFilter class</i>	34
6.4.2	<i>ContentList class</i>	34
6.4.3	<i>DatabaseQuery class</i>	34
6.4.4	<i>DeviceFilter class</i>	34
6.5	Package <i>org.ocap.hn.profile.upnp</i>	35
6.5.1	<i>UPnPConstants interface</i>	35
6.6	Package <i>org.ocap.hn.security</i>	39
6.6.1	<i>NetAuthorizationHandler interface</i>	39
6.6.2	<i>NetSecurityManager class</i>	39
6.7	Package <i>org.ocap.hn.service</i>	40
6.7.1	<i>RemoteService interface</i>	40
6.8	Package <i>org.ocap.hn.recording</i>	40
6.8.1	<i>NetRecordingEntry interface</i>	40
6.8.2	<i>NetRecordingRequestManager</i>	41
6.8.3	<i>NetRecordingRequestHandler interface</i>	42
6.8.4	<i>RecordingNetModule interface</i>	44
6.8.5	<i>RecordingContentItem interface</i>	46
6.9	Package <i>javax.tv.selection</i>	47
6.9.1	<i>ServiceContext Interface</i>	47
6.10	Package <i>org.javax.media (JMF)</i>	49
6.10.1	<i>Clock interface</i>	49
6.10.2	<i>Controller interface</i>	49
6.10.3	<i>Player interface</i>	50
ANNEX A XML DOCUMENTS (NORMATIVE)		51
A.1	OCAP Root Device Description.....	51
A.2	OC-DMS Device Description.....	51
ANNEX B REMOTE PLAYBACK WITH TIME-SHIFT		53
B.1	HTTP Push Model for MPEG Content.....	53
B.1.1	<i>Normal Play</i>	53
B.1.2	<i>Trick Modes</i>	53
B.2	HTTP Pull Model for MPEG Content.....	53
B.2.1	<i>Normal Play</i>	53
B.2.2	<i>Trick Modes</i>	54
ANNEX C OCAP NAME SPACE		55
C.1	Properties.....	55
C.1.1	<i>RecordingContentItem CDS object property definitions</i>	55
C.1.2	<i>Definitions of SRS properties</i>	61
C.1.3	<i>Definitions of OCAP-extended properties of CDS item class for series recordings</i>	63
C.2	Actions.....	63
C.2.1	<i>Recording X_PrioritizeRecordings</i>	63
ANNEX D CONTENT TSPEC POPULATION REQUIREMENTS		65
D.1	Default TSPEC values for Content Item.....	65
APPENDIX I COMPLEX MAPPING DIAGRAMS (INFORMATIVE)		66
I.1	Scheduling a Recording Sequence.....	67
I.2	Cancel Individual Recording	70
I.3	Stop Individual Recording.....	71

I.4	Delete Individual Recording.....	73
I.5	Schedule Local Recording.....	73
I.6	Add Individual Recording to Series Recording.....	75
APPENDIX II REVISION HISTORY		78

Figures

Figure 5-1	- OCAP Home Networking Framework for UPnP.....	7
Figure 5-2	- UPnP Device Architecture Protocol Stack	8
Figure 5-3	- Signal Flow Example.....	19
Figure 6-1	- ServiceContext States during RemoteService Presentation.....	48
Figure I-1	- Recording Schedule Sequence Diagram 1.....	67
Figure I-2	- Cancel Individual Recording Sequence Diagram.....	70
Figure I-3	- Stop Individual Recording Diagram.....	71
Figure I-4	- Delete Individual Recording Diagram.....	73
Figure I-5	- Schedule Local Recording Diagram.....	73
Figure I-6	- Add Recording to Series Diagram.....	75

Tables

Table 5-1	- HTTP Header Requirements	12
Table 5-2	- Mandatory Media Formats	17
Table 6-1	- TransferStatus Return Values	24
Table 6-2	- Recording Properties.....	45
Table C-1	- Recording Request Properties Mapped to CDS	55
Table C-2	- Recording Request State Mapping	56
Table C-3	- Arguments for X_PrioritizeRecordings().....	64
Table C-4	- Error Codes for X_Prioritize().....	64
Table D-1	- Default TSPEC parameter values.....	65

This page left blank intentionally.

1 SCOPE

1.1 Introduction and Overview

The OpenCable Home Networking Protocol defines a profile based on the Host Home Networking 2.0 Extension [HOST HN2] and the OCAP Home Networking Extension [OCAP HN]. This profile is independent of the physical network implementation, but runs over Internet Protocol (IP). It is designed to provide interoperability between OpenCable devices and a common protocol layer on a home network.

The UPnP protocol layer mapping is defined in Section 6. Alternative protocol layers may be added as sequential sections immediately following Section 6.

1.2 Purpose of document

This specification defines minimum technical requirements and additional features that must be implemented in order to provide a mapping to a protocol layer service over a home network.

1.3 Requirements

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

"SHALL"	This word means that the item is an absolute requirement of this specification.
"SHALL NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
"SHOULD NOT"	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
"MAY"	This word means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

2 REFERENCES

2.1 Normative References

In order to claim compliance with this specification, it is necessary to conform to the following standards and other works as indicated, in addition to the other requirements of this specification. Notwithstanding, intellectual property rights may be required to use or implement such normative references.

- [DLNAe2] DLNA 1.5 Guidelines Errata Volume 2, August 14, 2007.
- [DLNA vol 1] Digital Living Networking Alliance Home Networked Device Interoperability Guidelines; expanded: October 2006, Volume 1: Architectures and Protocols.
- [DLNA vol 2] Digital Living Networking Alliance Home Networked Device Interoperability Guidelines; expanded: October 2006, Volume 2: Media Format Profiles.
- [DLNA vol 3] Digital Living Networking Alliance Home Networked Device Interoperability Guidelines, expanded: October 2006, Volume 3: Link Protection.
- [HN-SEC] OpenCable Home Networking Security Specification, OC-SP-HN-SEC-I01-091217, December 17, 2009, Cable Television Laboratories, Inc.
- [HOST 2.1] OpenCable Host Device 2.1 Core Functional Requirements, OC-SP-HOST2.1-CFR-I11-100507, May 7, 2010, Cable Television Laboratories, Inc.
- [HOST HN2] Host Home Networking 2.0 Extension, OC-SP-HOST-HN2.0-I04-100507, May 7, 2010, Cable Television Laboratories, Inc.
- [MPEG2-1] ISO/IEC 13818-1:2000 Information technology -- Generic coding of moving pictures and associated audio information: Systems, International Standards Organization.
- [OCAP DVR] OCAP DVR Extension, OC-SP-OCAP-DVR-I06-100603, June 3, 2010, Cable Television Laboratories, Inc.
- [OCAP HN] OCAP Home Networking Extension, OC-SP-OCAP-HNEXT-I05-100603, June 3, 2010, Cable Television Laboratories, Inc.
- [OCAP] OpenCable Application Platform Specification (OCAP) 1.1 Profile, OC-SP-OCAP1.1.3-100603, June 3, 2010, Cable Television Laboratories, Inc.
- [RFC 2045] IETF RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, N. Freed, N. Borenstein, November 1996.
- [RFC 3986] IETF RFC 3986: Uniform Resource Identifiers (URI): Generic Syntax, T. Berners-Lee, MIT/LCS, R. Fielding, U.C. Irvine, L. Masinter, Xerox Corporation, January 2005.
- [RSD PROT] OpenCable Home Networking – Reserved Services Domain Protocol Specification, OC-SP-RSD-PROT-I01-080821, August 21, 2008, Cable Television Laboratories, Inc.
- [SCTE 43] ANSI/SCTE 43 2005, Digital Video Systems Characteristics for Cable Television.
- [UPNP AVTS] UPnP AVTransport v2, AVTransport:2 Service Template Version 1.01, UPnP Forum, May 31, 2006.
- [UPNP CDS] UPnP ContentDirectory v3, ContentDirectory:3 Service Template Version 1.01, UPnP Forum, September 30, 2008.
- [UPNP CMS] UPnP ConnectionManager v2, ConnectionManager:2 Service Template Version 1.01, UPnP Forum, May 31, 2006.

- [UPNP DA] UPnP Device Architecture Version 1.0: UPnP Device Architecture Specification Version 1.0.1, UPnP Forum, July 20, 2006.
- [UPNP MS] UPnP MediaServer v2, MediaServer:2 Device Template Version 1.01, UPnP Forum, May 31, 2006.
- [UPNP SRS] UPnP ScheduledRecording v1, ScheduledRecording:1 Service Template Version 1.01, UPnP Forum, September 19, 2007.

2.2 Informative References

This specification uses the following informative references:

- [DC] Dublin Core Metadata Element Set, Version 1.1.
- [DIDL-L] Digital Item Declaration Language – Lite.
- [HOSTHN] OpenCable Host 2.0 Home Networking Extension, OC-SP-HOST2-HNEXT-I01-060630, June 30, 2006, Cable Television Laboratories, Inc.
- [ISO 8601] ISO 8601:2000 Data elements and interchange formats – Information interchange -- Representation of dates and times, International Standards Organization, December 21, 2000.
- [MPEG1] ISO/IEC 11172-3:1993 Information technology -- Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s -- Part 3: Audio (MPEG-1Audio), International Standards Organization.
- [MPEG2-11] ISO/IEC 13818-11:2004 Information technology -- Generic coding of moving pictures and associated audio information – Part 11:IPMP on MPEG-2 Systems, International Standards Organization.
- [MPEG2-2] ISO/IEC 13818-2:2000 Information technology -- Generic coding of moving pictures and associated audio information: Video, International Standards Organization.
- [MPEG2-3] ISO/IEC 13818-3:1998 Information technology -- Generic coding of moving pictures and associated audio information: Audio, International Standards Organization.
- [PNG] W3C PNG Recommendations: Portable Network Graphics (PNG) Specification (Second Edition) Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification, ISO/IEC 15948:2003 (E), W3C, November 10, 2003.
- [RFC 1812] IETF RFC 1812: Requirements for IP version 4 Routers, F. Baker, June 1995.
- [RFC 1945] IETF RFC 1945: Hypertext Transfer Protocol – HTTP/1.0, T. Berners-Lee, MIT/LCS, R. Fielding, UC Irvine, H. Frystyk, May 1996.
- [RFC 2145] IETF RFC 2145, Use and Interpretation of HTTP Version Numbers, J. C. Mogul, DEC, R. Fielding, UC Irvine, J. Gettys, DEC, H. Frystyk, MIT/LCS, May 1997.
- [RFC 2326] IETF RFC 2326: Real Time Streaming Protocol (RTSP), H. Schulzrinne, Columbia U. A. Rao, Netscape, R. Lanphier, RealNetworks, April 1998.
- [RFC 2616] IETF RFC 2616: Hypertext Transfer Protocol – HTTP/1.1, R. Fielding, UC Irvine, J. Gettys, Compaq/W3C, J. Mogul, Compaq, H. Frystyk, W3C/MIT, L. Masinter, Xerox, P. Leach, Microsoft, T. Berners-Lee, June 1999.
- [RFC 3003] IETF RFC 3003: The audio/mpeg Media Type, M. Nilsson, November 2000.
- [RFC 3066] IETF RFC 3066: Tags for the Identification of Languages, H. Alvestrand, January 2002.
- [RFC 3551] IETF RFC 3551/STD0065: RTP Profile for Audio and Video Conferences with Minimal Control, H. Schulzrinne and S. Casner, July 2003.

- [RFC 3555] IETF RFC 3555: MIME Type Registration of RTP Payload Formats, S. Casner, Packet Design, P. Hoschka, July 2003.
- [RFC 3927] IETF RFC 3927: Dynamic Configuration of IPv4 Link-Local Addresses, May 2005.
- [RFC 4248] IETF RFC 4248 The telnet URI Scheme. P. Hoffman. October 2005.
- [RFC 4266] IETF RFC 4266 The gopher URI Scheme. P. Hoffman. November 2005.
- [SCTE 54] SCTE 54 2006, Digital Video Service Multiplex and Transport System Standard for Cable Television, Society of Cable Telecommunications Engineers Inc.
- [UPnP QMS] UPnP QoS Manager Service 3.0, Service Template Version 1.01, UPnP Forum, March 20, 2008.
- [XML] XML W3C Recommendations: Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, October 6, 2000.
- [XML NS] Namespaces in XML 1.0 (Second Edition), W3C Recommendation, 16 August 2006.

2.3 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027; Phone 303-661-9100; Fax 303-661-9199; Internet: <http://www.cablelabs.com/>
- Digital Living Networking (DLNA), Internet: <http://www.dlna.org/home>
- Dublin Core Metadata Initiative®, www.dublincore.org/documents/dces/
- European Telecommunications Standards Institute, Internet: <http://www.etsi.org/home.htm>
- International Organization for Standardization (ISO/IEC), Internet: <http://www.iso.org/iso/en/prods-services/ISOstore/store.html>
- Internet Engineering Task Force (IETF), Internet: <http://www.ietf.org>
- Society of Cable Telecommunication Engineers (SCTE), Internet: <http://www.scte.org/standards>
- UPnP, Internet: <http://www.upnp.org/>
- World Wide Web Consortium (W3C), Internet: www.w3.org

3 TERMS AND DEFINITIONS

This specification uses the following terms:

- HNIMP** An OCAP implementation that provides a network interface compliant with the Home Networking Interface Mapping Protocol (this specification).
- OC-DMP** OpenCable Digital Media Player. When a Host device is in the DMP device class as defined by this specification and complies with the requirements for an HNIMP in this specification, then it meets the criteria for an OC-DMP.
- OC-DMS** OpenCable Digital Media Server. When a Host device is in the DMS device class as defined by this specification and complies with the requirements for an HNIMP in this specification, then it meets the criteria for an OC-DMS.

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations and acronyms.

CA	Conditional Access
CCI	Copy Control Information
CCIF	CableCARD Interface
CDS	Content Directory Service
CMS	ConnectionManager Service
CSV	Comma Separated Value list; see [UPNP CDS] for definition
DIDL-L	Digital Item Declaration Language – Lite
DMS	Digital Media Server
EDN	Event Detection and Notification
GENA	General Event Notification Architecture
GOP	Group Of Pictures
HN Host	Home Networking Host
HTTP	Hypertext Transfer Protocol
IFO	InFOrmation (file extension format for DVD burning)
JMF	Java Media Framework
JPEG	Joint Photographic Experts Group
LPCM	Linear Pulse Code Modulation
MPEG	Moving Picture Experts Group
OCAP	OpenCable Application Platform
OCAPHN	OCAP Home Networking Extension
PID	Packet Identifier
PNG	Portable Network Graphics
PTS	Presentation Time Stamps; any reference to PTS is in compliance with [MPEG2-1]
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol
TSB	Time Shift Buffer
TSPEC	Traffic Specification
UPnP	Universal Plug-n-Play
URI	Uniform Resource Identifier
UTF-8	Universal Transformation Format - 8
XML	eXtensible Markup Language

5 TECHNICAL REQUIREMENTS

5.1 UPnP Protocol

Figure 5–1 shows an overview of the various layers that make up the OpenCable Home Networking Architecture. The OCAP HN APIs provide a high level abstraction of functionality that is independent of the underlying transport and network protocols and specific physical layer mechanism. The OpenCable HN Architecture maps the OCAP HN APIs to the UPnP Device Architecture/UPnP AV Architecture in order to provide device discovery and control, media management, and media transport functions. The UPnP Architecture itself is built on the standard TCP/UDP over IPv4 and IPv6 suite of protocols.

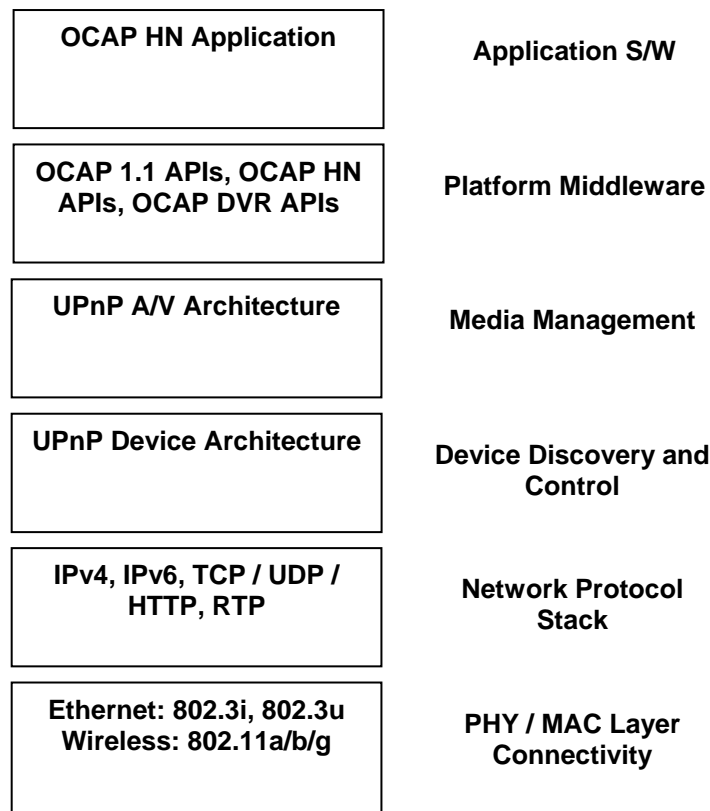


Figure 5–1 - OCAP Home Networking Framework for UPnP

Figure 5–2 shows the UPnP Device architecture stack in more detail.

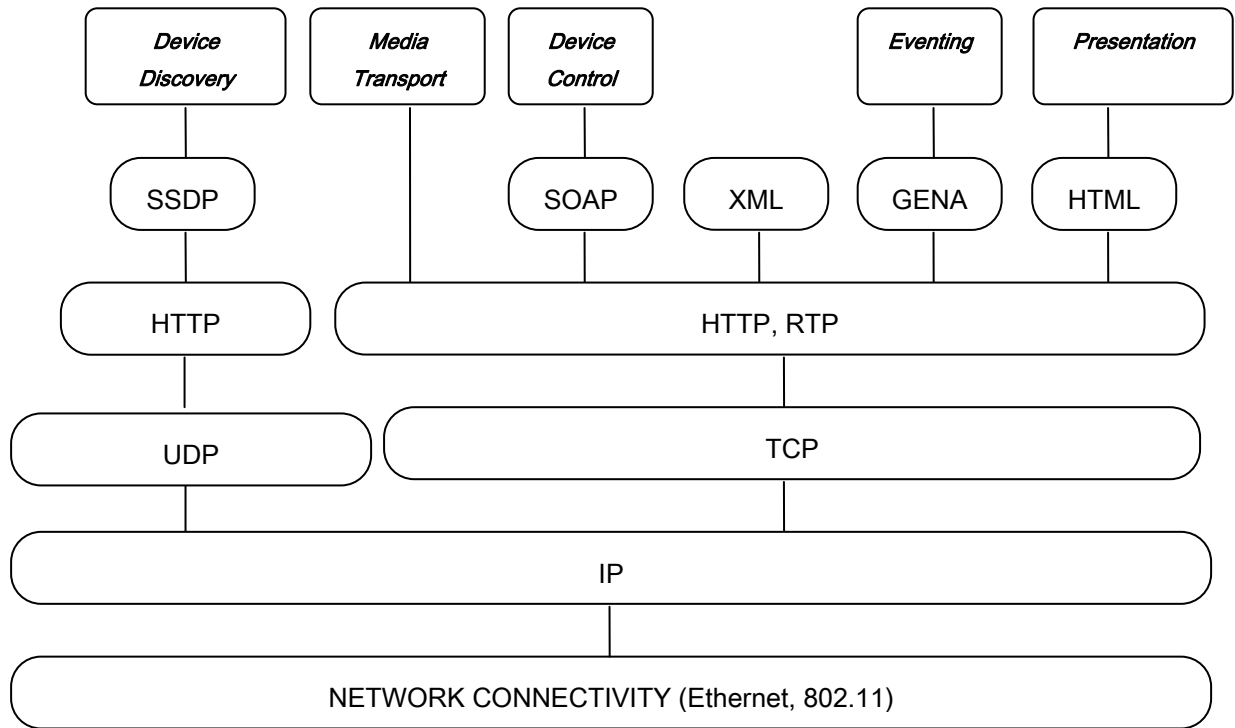


Figure 5–2 - UPnP Device Architecture Protocol Stack

5.2 DLNA Compliance

The HNIMP definition is based on the Digital Living Networking Alliance (DLNA) specifications. DLNA clarifies the use of the UPnP architecture, content formats, protocols, and so forth. This specification makes HNIMP requirements based on DLNA and adds or modifies requirements as appropriate for OpenCable device types. Where the DLNA Guidelines [DLNA vol 1] is referenced, the DLNA Guidelines Errata [DLNAe2] SHALL apply.

An OCAP Set-top or Terminal device, as defined by [HOST 2.1] specification, or later versions of that specification, is considered to be in the HND device category as defined by [DLNA vol 1] section 5.4 Device Categories.

When an HNIMP device supports playback of content received from a Home network, the device SHALL be considered to be in the DMP device class as defined by [DLNA vol 1] section 5.5 Device Classes and Roles.

When an HNIMP device supports transfer of local content to other devices on a Home network, the device SHALL be considered to be in the DMS device class as defined by [DLNA vol 1] section 5.5 Device Classes and Roles.

When an HNIMP supports the DMS device class and [OCAP DVR] is supported, the HNIMP SHALL support the aspects of the Scheduled Recording Service (SRS) identified in Section 6. SRS actions required by this specification SHALL comply with [UPNP SRS].

This specification extends [DLNA vol 1] references, and any [DLNA vol 1] reference to [59] AVTransport SHALL be read as a [UPNP AVTS] reference as defined in Section 2.1.

This specification extends [DLNA vol 1] references, and any [DLNA vol 1] reference to [60] ConnectionManager SHALL be read as a [UPNP CMS] reference as defined in Section 2.1.

This specification extends [DLNA vol 1] references, and any [DLNA vol 1] reference to [61] ContentDirectory SHALL be read as a [UPNP CDS] reference as defined in Section 2.1.

This specification extends [DLNA vol 1] references, and any [DLNA vol 1] reference to [62] Device Architecture SHALL be read as a [UPNP DA] reference as defined in Section 2.1.

This specification extends [DLNA vol 1] references, and any [DLNA vol 1] reference to [64] MediaServer SHALL be read as a [UPNP MS] reference as defined in Section 2.1.

5.3 Networking and Connectivity

When the HNIMP supports a hardware interface type defined by [DLNA vol 1] section 7.1 Networking and Connectivity, the HNIMP SHALL adhere to requirements from that section that apply to the interface type.

The HNIMP SHALL adhere to protocol requirements defined in [DLNA vol 1] section 7.1 Networking and Connectivity.

5.4 Device Discovery and Control

The HNIMP SHALL adhere to [DLNA vol 1] section 7.2 Device Discovery and Control for requirements that correspond to DLNA device classes supported by the implementation.

When an HNIMP supports the DMS device class, it SHALL adhere to the OC-DMS Device XML Template in Annex A.1.

NOTE: The above requirement on HNIMP is about compliance to the format of the Template in Annex A.2. Section 5.2 identifies requirements about services that need to be implemented by the HNIMP. The HNIMP populates only the implemented services in the OC-DMS device description.

The HNIMP SHALL provide an implementation of a UPnP Control Point as defined in [UPNP DA] for discovery and control of UPnP devices on the LAN.

5.4.1 Eventing

5.4.1.1 Last Change Events

LastChange events from a CDS, see [UPNP CDS], can be listened for by an application using a ContentServerNetModule when the represented CDS supports the LastChange state variable. A control point can determine if the LastChange state variable is supported if it is present in the service description. When an HNIMP subscribes to a CDS LastChange event, it is implementation-specific. However, if an HNIMP is not subscribed to LastChange events when an application calls ContentServerNetModule.addContentServerListener method and the service supports the LastChange state variable, then the HNIMP SHALL subscribe to LastChange events for that service at that time. In that case, the addContentServerListener method MAY return before the subscription completes. An HNIMP SHALL maintain a LastChange subscription as long as there are registered listener applications.

When CDS LastChange events are received corresponding to a ContentServerNetModule an application is listening to, the HNIMP SHALL generate one or more ContentServerEvent objects corresponding to the CDS event message and pass them to the corresponding ContentServerListener. When an HNIMP receives a LastChange message, it SHALL generate one ContentServerEvent object for all objAdd entries, another for all objMod entries, and another for all objDel entries in the LastChange message.

5.4.1.2 SystemUpdateID Events

SystemUpdateID events from a CDS, see [UPNP CDS], can be listened for by an application using the NetModule. When an HNIMP subscribes to a CDS SystemUpdateID event, it is implementation-specific. However, if an HNIMP is not subscribed to SystemUpdateID events when an application calls the NetModule.addNetModuleEventListener method, then the HNIMP SHALL subscribe to SystemUpdateID events for that service at that time. In that case, the addNetModuleEventListener method MAY return before the subscription completes. An HNIMP SHALL maintain a SystemUpdateID subscription as long as there are registered listener applications.

When SystemUpdateID events are received corresponding to a CDS NetModule (ContentServerNetModule) an application is a listener to, the HNIMP SHALL generate a NetModuleEvent with STATE_CHANGE type that corresponds to the SystemUpdateID message.

5.5 Media Management

The HNIMP SHALL adhere to [DLNA vol 1] section 7.3 Media Management for requirements that correspond to device types supported by the implementation.

The [DLNA vol 1] specification section 7.3.84.1 is extended and an HNIMP SHALL support BCM. The [DLNA vol 1] specification section 7.4.54.3 is extended and an HNIMP that is an HTTP client endpoint SHALL use a scid.dlna.org header in subsequent HTTP requests to identify a known and associated UPNP AV Connection only if the HTTP requests are for the same URI.

An HNIMP OC-DMS SHALL support the following CDS actions:

- UpdateObject
- DeleteResource
- DestroyObject
- Search

An HNIMP that implements a ContentDirectory service SHALL implement the Tracking Changes Option as defined by [UPNP CDS] section 2.2.10.

When the HNIMP supports the MediaRenderer service, it SHALL support volume control. See [DLNA vol 1] requirement 7.3.108 for further requirements implications.

When a upnp:channelID or srs:scheduledChannelID is used and the type is SI, the fields are populated as follows:

- Network ID not used
- Transport stream ID not used
- Service ID SHALL be equal to the source_id signaled in a VCT

5.5.1 Device Namespace

OCAP HN devices SHALL employ the <ocap:X_OCAPHN> XML element inside the <device> element of the device description document to indicate adherence to a particular OCAP Home Networking Protocol document version. The value of this element is the DLNA device class, a dash character, followed by the numeric version value of the document. The <ocap:X_OCAPHN > element indicates compliance for a specific <device>, excluding its embedded devices listed in <deviceList>. An example of <ocap:X_OCAPHN > element is shown as follows:

```
<ocap:X_OCAPHN xmlns:ocap="urn:schemas-cablelabs-com:device-1-0">OC-DMS-1.0</ocap:X_OCAPHN>
```

The "urn:schemas-cablelabs-com:device-1-0" namespace SHALL be specified for the <ocap:X_OCAPHN > element, and the namespace prefix SHALL be "ocap:". The namespace prefix declaration MAY be specified in the <root> element of the device description. The value of the <ocap:X_OCAPHN > element is a string as defined below. Linear white spaces (LWS) are not implied in this definition below.

```
ocaphn-value = ocaphn-dev-class "-" ocaphn-version
ocaphn-dev-class = "DMS" (DMS = Digital Media Server)
ocaphn-version = major-version "." minor-version
major-version = DIGIT
minor-version = DIGIT
```

5.5.2 Metadata Namespace

For the ocap properties defined in Annex C populated in the DIDL-Lite document, the HNIMP SHALL use the following namespace: "urn:schemas-cablelabs-org:metadata-1-0/", and the HNIMP SHALL use "ocap" as namespace prefix for these properties. This namespace is not defined through a formal XML schema. This allows definition of new XML attributes in the future without having to define a new namespace or schema.

This requirement standardizes the "ocap" namespace so that it can be placed anywhere in the DIDL-Lite document.

5.5.3 Middleware Elements

OCAP HN devices SHALL employ the <ocap:X_MiddlewareProfile> XML element inside the <device> element of any root device description document to indicate the OCAP profile, e.g., 1.0.2. The value of this element matches the OCAP profile version supported, e.g., "1.0.2". An example of <ocap:X_MiddlewareProfile> element is shown as follows:

```
<ocap:X_MiddlewareProfile xmlns:ocap="urn:schemas-cablelabs-com:device-1-0">1.0.2</ocap:X_MiddlewareProfile>
```

OCAP HN devices SHALL employ the <ocap:X_MiddlewareVersion> XML element inside the <device> element of any root device description document to indicate the vendor-specific OCAP implementation version. An example of <ocap:X_MiddlewareVersion> element is shown as follows:

```
<ocap:X_MiddlewareVersion xmlns:ocap="urn:schemas-cablelabs-com:device-1-0">1a.2.3</ocap:X_MiddlewareVersion>
```

5.6 Media Transport

The HNIMP SHALL adhere to all requirements in [DLNA vol 1] section 7.4 Media Transport for requirements that correspond to device types supported by the implementation.

When an HNIMP is a DMS device, it SHALL support the TimeSeekRange.dlna.org HTTP header field; see [DLNA vol 1] section 7.4.40.

When an HNIMP is a DMS device, it SHALL support the PlaySpeed.dlna.org HTTP header field; see [DLNA vol 1] section 7.4.70.

When an HNIMP is a DMS device, it SHALL support the Range HTTP header field for content to which Link Protection is not applied; see [DLNA vol 1] section 7.4.38.

When an HNIMP is a DMS device, it SHALL support the Range.dtcp.com HTTP header field for content to which DTCP-IP Link Protection is applied; see [DLNA vol 3] section 8.4.1.

When serving streaming video, the HNIMP SHALL NOT exceed a maximum bitrate of (1.5 * normal play speed bitrate of the media format) based on peak bitrate output measurement, regardless of trick mode speed and QoS configuration. Normal play speed bitrates are defined in [DLNA vol 2] as bitrates for specific media formats.

If a `FrameRateInTrickMode` header (Section 5.6.1.8) is provided by the client, but the `ServersidePacedStreaming` header (Section 5.6.1.6) is not provided by the client to perform trick modes, then the HNIMP server is not required to re-stamp Presentation Time Stamps (PTS) in the MPEG-2-TS. In this case, the PTS will be decrementing during the rewind operation. During trick mode operations, the HNIMP client devices SHALL be able to playback streams with decrementing PTS.

If a `FrameRateInTrickMode` header (Section 5.6.1.8) is provided by the client, but the `ServersidePacedStreaming` header (Section 5.6.1.6) is not provided by the client to perform trick modes, then the HNIMP server is not required to re-stamp Program Reference Clock (PCR) in the MPEG-2-TS.

When the HNIMP re-stamps PTS, it SHALL re-stamp the corresponding PCR. If PTS is not re-stamped, the HNIMP SHALL NOT re-stamp the corresponding PCR.

The HNIMP client devices SHALL be able to playback content streams whether or not PTS and PCR re-stamping is performed by the HNIMP server during trick mode operations.

Note (Informative): During the trick mode operation, the HNIMP server is not required to maintain the same frame rate as for the normal playback.

During the fast forward operation using HTTP transport on content stream that is currently being recorded, the HNIMP Server SHALL close the HTTP connection with the client when the server reaches a live point in the content stream. The HNIMP server SHALL NOT close the connection until it has sent all the content up to the live point in the content stream.

When a client HNIMP supports RTP, it SHOULD NOT request Wall Clock Time Samples; see [DLNA vol 1] requirement 7.4.117.1.

When an HNIMP supports RTP and a PLAY request includes the Supported header with the feature tag `dlna.announce`, then the Serving Endpoint SHALL send an ANNOUNCE request to the Receiving Endpoint when the last RTP packet has been sent. This is recommended by [DLNA vol 1] requirement 7.4.261.1.

5.6.1 HTTP Headers

HNIMP rendering devices that use scaled network traffic for trick modes when streaming video SHOULD maintain the jitter characteristics of local recording rendering. This specification defines a number of HTTP headers in addition to [DLNA vol 1] in the `ochn.org` name space that add characteristics to video streaming in order to assist with media presentation and reduce noticeable anomalies. When encountering syntax errors with any HTTP headers in the `ochn.org` name space, the HTTP server SHALL use the HTTP response code 400 (Bad Request). When encountering an HTTP header in the `ochn.org` name space that is not supported, the HTTP server SHALL use the HTTP response code 406 (Not Acceptable).

The table below lists the headers, indicates direction from client to server, the server support requirement, and, if the header is used in trick play, normal play or both. When the header direction is bi-directional, it begins with a client request.

Table 5–1 - HTTP Header Requirements

Header	Direction	Server support	Trick/Normal
<code>MaxTrickModeBandwidth.ochn.org</code>	Client→Server	optional	Trick
<code>CurrentDecodePTS.ochn.org</code>	Client→Server	optional	Trick
<code>ChunkEncodingMode.ochn.org</code>	Client→Server	required	Trick
<code>MaxGOPsPerChunk.ochn.org</code>	Client↔Server	required	Trick

Header	Direction	Server support	Trick/Normal
MaxFramesPerGOP.ochn.org	Client→Server	optional	Trick
ServersidePacedStreaming.ochn.org	Client→Server	optional	Trick/Normal
FrameTypesInTrickMode.ochn.org	Client↔Server	required	Trick
FrameRateInTrickMode.ochn.org	Client↔Server	required	Trick
PresentationTimeStamps.ochn.org	Server→Client	optional	Trick/Normal
AVStreamParameters.ochn.org	Server→Client	optional	Trick/Normal

5.6.1.1 MaxTrickModeBandWidth

An HNIMP OC-DMS MAY support the MaxTrickModeBandWidth.ochn.org header. This header SHALL NOT be included in any response. The definition of this header is it requests the maximum peak bitrate at the socket layer requested by a client that the server complies with. Physical bitrates may be slightly higher with Ethernet IP and TCP headers. The notation for the MaxTrickModeBandWidth.ochn.org header is defined as follows:

- MaxTrickModeBandWidth-line = "MaxTrickModeBandWidth.ochn.org" *LWS ":" *LWS maximum trick mode bandwidth
- maximum trick mode bandwidth = "bandwidth" "=" bps
- bps = <the numerical bits per second value, cannot be negative and cannot exceed the maximum bitrate for video streaming>

Note that the "bandwidth" token is case sensitive.

Example:

- MaxTrickModeBandWidth.ochn.org : bandwidth=6291456

5.6.1.2 CurrentDecodePTS

An HNIMP OC-DMS MAY support the CurrentDecodePTS.ochn.org header. This header SHALL NOT be included in any response. The definition of this header is the renderer current decoder timestamp in 32 bit hex format based on the 45 KHz clock. The notation for the CurrentDecodePTS.ochn.org header is defined as follows:

- CurrentDecodePTS -line = "CurrentDecodePTS.ochn.org" *LWS ":" *LWS current decoder PTS
- current decoder PTS = "PTS" "=" pts
- pts = 8 hexdigit
- hexdigit = <hexadecimal digit: "0"-"9", "A"-"F", "a"-"f">

Note that the "PTS" token is case sensitive.

Example:

- CurrentDecodePTS.ochn.org : PTS=00070800

5.6.1.3 ChunkEncodingMode

An HNIMP OC-DMP MAY use the ChunkEncodingMode header to request Chunked Transfer Coding as per [DLNA vol 1]. The ChunkEncodingMode.ochn.org header SHALL be supported by an HNIMP OC-DMS. Defined as an indication for chunked encoding type where type can be GOP, Frame, or other.

The "Frame" chunk type SHALL be used in conjunction with the frame types defined in the FrameTypesInTrickMode.ochn.org header. The ChunkEncodingMode header with "Frame" chunk type SHALL be ignored if the FrameTypesInTrickMode header is not present in the HTTP request. When the ChunkEncodingMode header is included in a normal playback HTTP request, i.e., the PlaySpeed.dlna.org header is not present or the PlaySpeed is set to 1, the OC-DMS SHALL also ignore this header request.

The notation for the ChunkEncodingMode.ochn.org header is defined as follows:

- ChunkEncodingMode -line = "ChunkEncodingMode.ochn.org" *LWS ":" *LWS chunk type
- chunk type : "chunk" "=" <"GOP" | "Frame" | "Other">

Note 1: the "chunk" token is case sensitive.

Note 2: "Other" MAY include for example; crypto-chunk, where the chunk contents are server-specific.

Example:

- ChunkEncodingMode.ochn.org : chunk="GOP"

When OC-DMS receives a request with this header with PlaySpeed different than 1, the OC-DMS SHALL send the response using Chunked Transfer Coding where each chunk is made with specified unit. Note that when the OC-DMS sends out only I-frame in trick mode, "chunk=GOP" means each chunk be made of one I-frame only.

5.6.1.4 MaxGOPsPerChunk

An HNIMP OC-DMP MAY use the MaxGOPsPerChunk header to specify the maximum number of GOPs per chunk when in trick mode. This header MAY be ignored if the current mode is not chunked encoding or the play rate speed is 1. An HNIMP OC-DMS SHALL support the MaxGOPsPerChunk.ochn.org header. The definition of this header is the maximum number of groups of pictures per chunk. The notation for the MaxGOPsPerChunk.ochn.org header is defined as follows:

- MaxGOPsPerChunk-line = "MaxGOPsPerChunk.ochn.org" *LWS ":" *LWS number of GOPs
- number of GOPs = "gops" "=" gops
- gops = 3 decdigit
- decdigit = <decimal digit: "0"-"9">

Note that the "gops" token is case sensitive.

Example:

- MaxGOPsPerChunk.ochn.org : gops=1

When OC-DMS receives the HTTP GET request with MaxGOPsPerChunk header from OC-DMP, the OC-DMS SHALL construct each chunk from a maximum of the number of GOPs as specified by this header, and the OC-DMS SHALL include this header set to the number of GOPs of each chunk. For example,

- When OC-DMP sends HTTP GET request with MaxGOPsPerChunk set to 1, the OC-DMS constructs each chunk from exactly one GOP.
- When OC-DMP sends HTTP GET request with this parameter set to 3, the OC-DMS constructs each chunk from at most three GOPs.

5.6.1.5 MaxFramesPerGOP

An HNIMP OC-DMP MAY use the MaxFramesPerGOP header to request the maximum number of frames per GOP in trick mode and has no affect unless current mode is chunked encoding. An HNIMP OC-DMS MAY support the MaxFramesPerGOP.ochn.org header. The definition of this header is the maximum number of frames per group of pictures in trick mode. The notation for the MaxFramesPerGOP.ochn.org header is defined as follows:

- MaxFramesPerGOP -line = "MaxFramesPerGOP.ochn.org" *LWS ":" *LWS number of frames
- number of frames = "frames" "=" frames
- frames = 3 decdigit
- decdigit = <decimal digit: "0"-"9">

Note that the "frames" token is case sensitive.

Example:

- MaxFramesPerGOP.ochn.org : frames=1

5.6.1.6 *ServersidePacedStreaming*

An HNIMP OC-DMP MAY use the ServersidePacedStreaming header to request a server to pace data based on time stamps in the content. An HNIMP OC-DMS MAY support the ServersidePacedStreaming.ochn.org header. The definition of this header is an indication to pace data according to content timestamps. The default is no pacing. The notation for the ServersidePacedStreaming.ochn.org header is defined as follows:

- ServersidePacedStreaming -line = "ServersidePacedStreaming.ochn.org" *LWS ":" *LWS pacing enabled
- pacing enabled : "pacing" "=" <"YES" | "NO">

Note that the "pacing" token is case sensitive.

Example:

- ServersidePacedStreaming.ochn.org : pacing=YES

5.6.1.7 *FrameTypesInTrickMode*

An HNIMP OC-DMP MAY use the FrameTypesInTrickMode header to request a server to respond with the frame types in trick mode value. An HNIMP OC-DMS SHALL support the FrameTypesInTrickMode.ochn.org header, i.e., OC-DMS SHALL select the frames whose type is specified by this header and SHALL include this header with the correct value. The definition of this header is a value returned from the server to indicate the frame types that can be expected in the next group of pictures. The notation for the FrameTypesInTrickMode.ochn.org header is defined as follows:

- FrameTypesInTrickMode -line = "FrameTypesInTrickMode.ochn.org" *LWS ":" *LWS frame types
- frame types = "frames" "=" <"I" | "IP" | "all">

Note that the frames token is case sensitive.

Example:

- FrameTypesInTrickMode.ochn.org : frames=I
- FrameTypesInTrickMode.ochn.org : frames=all

When the OC-DMP sends the HTTP-GET request which has this header with "frame=I", the OC-DMS SHALL include this header in the response with "frame=I" and SHALL thereafter send only I-frames.

When the OC-DMP sends the HTTP-GET request which has this header with "frame=IP", the OC-DMS SHALL include this header in the response with "frame=I" or "frame=IP" and SHALL thereafter send only I-frames or I and P frames, respectively.

When the OC-DMP sends the HTTP-GET request which has this header with "frame=all", the OC-DMS SHALL include this header in the response with "frame=I", "frame=IP" or "frame=all" and SHALL thereafter send only I-frames, I and P frames or all frames, respectively.

5.6.1.8 *FrameRateInTrickMode*

An HNIMP OC-DMP MAY use the `FrameRateInTrickMode` header to request a server to respond with the value for frame rate in trick mode. An HNIMP OC-DMS SHALL support the `FrameRateInTrickMode.ochn.org` header. The definition of this header is a value returned from the server to indicate the frames per second at the current play speed. OC-DMP sends this header without a value to request OC-DMS to report the frame rate in the response. OC-DMS MAY select the frame rate considering the current play speed, bandwidth limitation, etc. OC-DMS SHALL include this header with the value of selected frame rate in the response. The notation for the `FrameRateInTrickMode.ochn.org` header is defined as follows:

- `FrameRateInTrickMode` -line = "FrameRateInTrickMode.ochn.org" *LWS ":" *LWS frame rate
- frame rate = "rate" "=" rate
- rate = 2 decdigit
- decdigit = <decimal digit: "0"-"9">

Note that the "rate" token is case sensitive.

Example:

- `FrameRateInTrickMode.ochn.org` : rate=15

5.6.1.9 *PresentationTimeStamps*

An HNIMP OC-DMS MAY use the `PresentationTimeStamps` header in a response to a request for an MPEG stream of finite duration, in order to inform an OC-DMP what the first and last presentation time stamp values are for the stream. An OC-DMP MAY use this information to assist with presentation. The notation for the `PresentationTimeStamps.ochn.org` header is defined as follows:

- `PresentationTimeStamps` -line = " PresentationTimeStamps.ochn.org"*LWS"."*LWS startPTS *LWS endPTS
- startPTS="startPTS" "=" pts
- endPTS="endPTS" "=" pts
- pts = 8 hexdigit
- hexdigit = <hexadecimal digit: "0"-"9", "A"-"F", "a"-"f">

Note that the "startPTS" and "endPTS" tokens are case sensitive.

Note that values of the pts term are in 45 KHz units.

5.6.1.10 *AVStreamParameters*

An HNIMP OC-DMS MAY use the `AVStreamParameters` header to inform an OC-DMP of audio and video PID and stream type values for a connection that is being used to stream MPEG content. The PID and stream type values are for primary audio and video and do not include other streams, e.g. alternate languages. An OC-DMP MAY use this information to assist with presentation. The notation for the `AVStreamParameters.ochn.org` header is defined as follows:

- `AVStreamParameters` -line = " AVStreamParameters.ochn.org"*LWS"."*LWS videoPID *LWS videoType *LWS audioPID *LWS audioType
- videoPID = " videoPID" "=" decdigit
- videoType = " videoType" "=" decdigit
- audioPID = " audioPID" "=" decdigit

- audioType =" audioType" "=" decdigit
- decdigit=<decimal digit: "0"-"9">

Note that the "videoPID", "videoType", "audioPID", and "audioType" tokens are case sensitive.

Note that the videoType and audioType values SHALL adhere to stream type assignments defined in [MPEG2-1].

5.7 Content Transformation Device Virtualization

When the HNIMP implements a virtual server or renderer as defined by [DLNA vol 1] section 7.5, it SHALL adhere to all requirements in [DLNA vol 1] section 7.5 Content Transformation Device Virtualization.

5.8 Content Formats

An HNIMP DMP SHALL support the rendering of content formats as specified in [OCAP]. This includes both broadcast streaming and monomedia-based content formats. Additionally, an HNIMP DMP SHALL support the rendering of a subset of the content formats defined in [DLNA vol 2]. An HNIMP DMS SHALL support the same formats for streaming and download purposes. The required formats for home network support are described in Table 5–2.

Table 5–2 - Mandatory Media Formats

Media Type	OCAP	DLNA Profile ID
Images	JPEG, PNG, GIF	JPEG_SM max resolution 640 x 480, JPEG_MED, JPEG_LRG, JPEG_TN, JPEG_SM_ICO, JPEG_LRG_ICO, PNG_TN, PNG_SM_ICO, PNG_LRG_ICO, PNG_LRG, GIF_LRG
Video	MPEG-2 TS according to [SCTE 43]	MPEG_PS_NTSC (MPEG-2 PS), MPEG_PS_NTSC_XAC3, MPEG_TS_SD_NA_ISO, MPEG_TS_SD_NA_XAC3_ISO, MPEG_TS_HD_NA_ISO, MPEG_TS_SD_NA_T, MPEG_TS_HD_NA_T, MPEG_TS_HD_NA_XAC3_ISO, AVC_TS_MP_SD_MPEG1_L3_ISO, AVC_TS_MP_SD_AAC_MULT5_ISO, AVC_TS_MP_SD_AC3_ISO, AVC_TS_MP_HD_MPEG1_L3_ISO, AVC_TS_MP_HD_AAC_MULT5_ISO, AVC_TS_MP_HD_AC3_ISO
Monomedia Audio	MPEG-1 layer 1, 2 & 3; Dolby AC-3	LPCM

Media Type	OCAP	DLNA Profile ID
Broadcast Streaming Audio	Dolby AC-3	LPCM AC3, MP3, MP3X, AAC_ADTS, AAC_ADTS_320, AAC_ISO, AAC_ISO_320, AAC_LTP_ISO, AAC_LTP_MULT5_ISO, AAC_LTP_MULT7_ISO, AAC_MULT5_ADTS, AAC_MULT5_ISO

When content for any profile ID that begins with MPEG_TS or AVC_TS is being streamed using any network protocol such as HTTP, the server SHALL include PAT and PMT occurrences at rates defined by [MPEG2-1] with respect to the wall clock time. In addition, any data streams that are part of the source service SHALL be included in the output data stream. The HNIMP server SHALL include only one PMT entry in the PAT (i.e., content stream served by the HNIMP server needs to be a SPTS).

5.9 Quality of Service (QoS)

The HNIMP MAY support quality of service functionality as defined in [RSD PROT].

5.10 Content Protection

The current version of [HOSTHN] only allows for discovery and rendering of personal content as well as cable recorded services and prohibits the transport of cable-delivered broadcast content on the LAN regardless of the state of copy protection applied to the content. When cable recorded services are transported over a Home network, they SHALL be protected as defined in the [HN-SEC] specification. Later versions of this specification will address the issue of broadcast cable content on the LAN.

5.11 UPnP Interoperability

The HNIMP SHALL comply with UPnP interoperability guidelines as specified in [UPNP DA]. Figure 5–3 demonstrates a sample interaction between UPnP device discovery and an application discovering devices and then performing basic UPnP actions. Some of the UPnP actions depicted are optional, e.g., CM:PrepareForConnection.

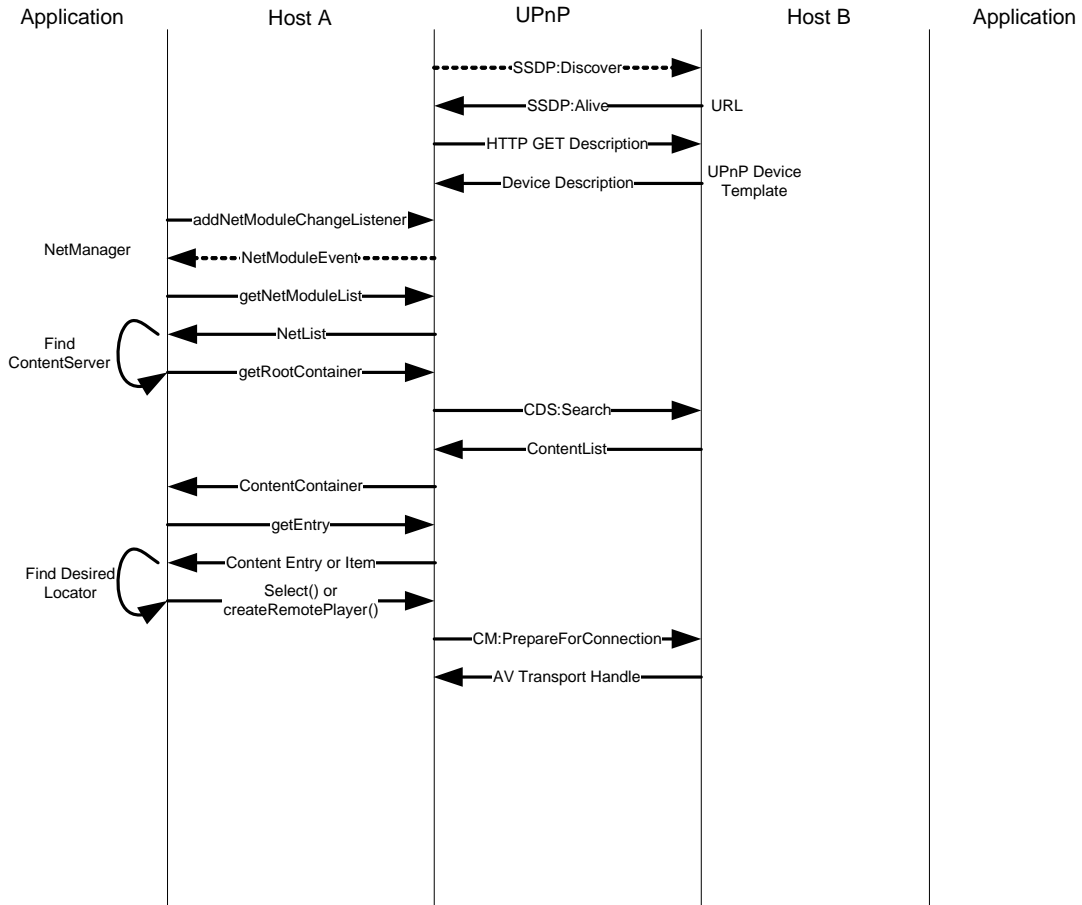


Figure 5-3 - Signal Flow Example

6 UPNP MAPPING REQUIREMENTS

OCAP Home Networking Extension-compliant devices [OCAP HN] implementing the UPnP protocol SHALL map OCAP Home Networking API methods to UPnP actions and properties, per this section. The same mappings apply whether the API object is created by an application or the HNIMP. In addition, the mappings apply to API objects that represent local UPnP objects.

6.1 Device Registry

OCAP Home Networking Extension-compliant devices [OCAP HN] implementing the UPnP protocol maintain a current list of devices and services based on received UPnP device available and device unavailable messages as specified in [UPNP DA]. The HNIMP SHALL maintain local devices and services in the device and network module lists returned by the HN API defined in [OCAP HN]. The HNIMP SHOULD maintain the latest device and service description information as defined by [UPNP DA] in order to respond to Device and NetModule method calls without blocking.

When the HNIMP receives a device description, it SHALL create a Device object for it and populate the property fields from the device description accordingly, e.g., manufacturer.

When the HNIMP receives a device description that contains a service description, it SHALL create a NetModule that coincides with service type. When a CDS description is received, the HNIMP SHALL create a ContentServerNetModule in the corresponding Device. When an SRS description is received, the HNIMP SHALL create a RecordingNetModule in the corresponding Device.

While a device subscription is in effect as described by [DLNA vol 1] requirements 7.2.21 and 7.2.22, the HNIMP SHALL generate NetModule events for services to which the control point is subscribed.

When a device subscription times out as defined by [DLNA vol 1] 7.2.21 and 7.2.22, the HNIMP control point SHALL NOT generate NetModule events for the timed out service.

When an HNIMP responds to a device discovery request it SHALL send "OCAP-HOST" or "OCAP-TERMINAL" as the root device. In addition, the HNIMP SHALL include "OC-DMS" as a sub-device, if supported.

The HNIMP SHALL create a org.ocap.hn.Device object for each remote device and sub-device discovered. The HNIMP SHALL create a org.ocap.hn.Device object for each local device transmitted in an M-SEARCH response. The HNIMP SHALL create a org.ocap.hn.Device object corresponding to an OC-DMP if supported.

The HNIMP MAY include each supported RSD device as a sub-device in UPnP device discovery; see [RSD PROT] for RSD device details.

6.2 Package org.ocap.hn

6.2.1 ContentServerNetModule interface

6.2.1.1 requestRootContainer method

The getRootContainer method SHALL use the ContentDirectory service Browse action with the following argument settings:

- ObjectID set to 0,
- BrowseFlag set to BrowseMetadata,

- StartingIndex set to 0,
- RequestedCount set to 0,
- SortCriteria set to ""

The ContentList result SHALL be returned via the corresponding NetActionEvent.getResponse method. It SHALL contain one entry, which is a ContentContainer representing the root container of the server. Mandatory OCAP and UPnP CDS properties that apply to the container SHALL be included in the response.

6.2.1.2 requestBrowseEntries method

The requestBrowseEntries method SHALL use the [UPnP CDS] Browse action. The action input arguments SHALL be mapped to the method parameters as follows:

- ObjectID set to the startingEntryID parameter,
- BrowseFlag set to BrowseDirectChildren when browseChildren parameter is true; otherwise it is set to BrowseMetadata,
- StartingIndex set to startingIndex parameter,
- Filter set to the propertyFilter parameter,
- RequestedCount set to requestedCount parameter,
- SortCriteria derived from the sortCriteria parameter if not null. If null, empty string is used.

The HNIMP receiving the action SHALL include all properties required by this specification for the object type in the returned result in addition to those properties specified in the propertyFilter parameter.

The ContentList result SHALL be returned via the corresponding NetActionEvent.getResponse method. Each entry in the list SHALL be a ContentEntry that maps to a content item or container in the CDS that was browsed. The HNIMP receiving the action response SHALL include all properties contained in the UPnP action response in the returned result.

6.2.1.3 requestSearchCapabilities method

The requestSearchCapabilities method SHALL use the [UPnP CDS] GetSearchCapabilities action. The ContentList result SHALL be returned via the corresponding NetActionEvent.getResponse method. It SHALL contain one entry, which is a String object matching the search criteria parameter format defined by the requestSearchEntries method description.

6.2.1.4 requestSearchEntries method

The requestSearchEntries method SHALL use the [UPnP CDS] Search action. The action input arguments SHALL be mapped to the method parameters as follows:

- ContainerID set to the parentID parameter,
- SearchCriteria set to the searchCriteria parameter. If the searchCriteria parameter contains a value of null, SearchCriteria SHALL be set to "*".
- Filter set to the propertyFilter parameter,
- StartingIndex set to startingIndex parameter,
- RequestedCount set to requestedCount parameter,
- SortCriteria derived from the sortCriteria parameter if not null. If null, empty string is used.

The HNIMP receiving the action SHALL include all properties required by this specification for the object type in the returned result in addition to those properties specified in the propertyFilter parameter. The ContentList result SHALL be returned via the corresponding NetActionEvent.getResponse method. Each entry in the list SHALL be a ContentEntry that maps to a content item or container in the CDS that was searched. The HNIMP receiving the action response SHALL include all properties contained in the UPnP action response in the returned result.

6.2.1.5 addContentServerListener method

See Section 5.4.1.1, Last Change Events.

6.2.1.6 removeContentServerListener method

See Section 5.4.1.1, Last Change Events.

6.2.2 Device interface

6.2.2.1 getVersion method

The getVersion method SHALL return the version value from the deviceType element found in the device's UPnP Device Description document. The version value is the last numerical term in the deviceType element.

6.2.2.2 getCapabilities method

The getCapabilities method SHALL return an enumeration of capabilities string constants defined in the Device interface. When the device is a server device, the getCapabilities method SHALL return the CAP_STREAMING_SUPPORTED constant. When the device contains an SRS service, the getCapabilities method SHALL return the CAP_RECORDING_SUPPORTED constant. When the device is an OC-DMS, the getCapabilities method SHALL return the CAP_REMOTE_STORAGE_SUPPORTED, as well as the CAP_TUNER_SUPPORTED constants. Any capability returned from a sub-device SHALL also be returned for its root device.

6.2.2.3 getName method

The getName method SHALL return the same value as the getProperty method when passed the PROP_FRIENDLY_NAME constant.

6.2.2.4 getProperty method

This method takes a property name string constant defined in the Device interface and returns a value mapped to the device discovery. Values returned from elements in each <device> element that is part of the root device or a sub-device and defined by the long description defined in [UPNP DA] section 2.1 SHALL be populated as follows:

- PROP_DEVICE_TYPE parameter returns value from the <deviceType> element.
- PROP_FRIENDLY_NAME parameter returns value from the <friendlyName> element.
- PROP_MANUFACTURER parameter returns value from the <manufacturer> element.
- PROP_MANUFACTURER_URL parameter returns value from the <manufacturerURL> element.
- PROP_MODEL_DESCRIPTION parameter returns value from the <modelDescription> element.
- PROP_MODEL_NAME parameter returns value from the <modelName> element.
- PROP_MODEL_NUMBER parameter returns value from the <modelNumber> element.
- PROP_MODEL_URL parameter returns value from the <modelURL> element.

- PROP_SERIAL_NUMBER parameter returns value from the <serialNumber> element.
- PROP_UDN parameter returns value from the <UDN> element.
- PROP_UPC parameter returns value from the <UPC> element.
- PROP_PRESENTATION_URL parameter returns value from the <presentationURL> element.

Values taken from the <device> element in the root device and returned for the Device representing a root device, as well as each Device representing a sub-device under the same root, SHALL be populated as follows:

- PROP_MIDDLEWARE_PROFILE parameter returns value from the <ocap:X_MiddlewareProfile> element.
- PROP_MIDDLEWARE_VERSION parameter returns value from the <ocap:X_MiddleVersion> element.
- The PROP_LOCATION value SHALL be returned from the LOCATION field of a search response defined in [UPNP DA] section 1.2.3 for a root device and each sub-device. This always provides the location URL of the root device.getType method.

This method returns one of the TYPE constants defined in the Device interface. The TYPE constant returned SHALL be determined by the <deviceType> element from the UPnP discovery message for the root or sub-device represented by the Device object this method is called in.

6.2.2.5 *getInetAddress*

The getInetAddress method SHALL return an object implementing an appropriate sub-interface of the java.net.InetAddress interface. This may be InetAddress or Inet6Address. This object SHALL represent the originating IP address of the device description documents received for this Device during device discovery.

6.2.2.6 *setFriendlyName method*

The setFriendlyName method modifies the <friendlyName> property within the <device> element of the HNIMP device description. The method SHALL throw IllegalArgumentException if the value parameter is greater than 63 characters or contains characters that are not valid for the language indicated by the ACCEPT-LANGUAGE or CONTENT-LANGUAGE headers. See the [UPNP DA] specification for device description definition and the [DLNA vol 1] specification for DLNA defined device properties.

When the value parameter is valid and different from the existing <friendlyName> property, the HNIMP SHALL adhere to the following steps in order:

1. Return from the method.
2. Send an ssdp:byebye message.
3. Update the <friendlyName> property using the value parameter.
4. Send an ssdp:alive message.
5. Generate device events to DeviceListener objects.

If the value parameter matches the <friendlyName> property, this method does nothing and returns successfully.

6.2.3 DeviceEvent class

When the HNIMP performs an M-SEARCH operation, it SHALL generate DeviceEvent and NetModuleEvent occurrences to registered listener applications as appropriate for the changes in configuration.

When a `ssdp:alive` message cache duration expires, the HNIMP SHALL generate `DeviceEvent` and `NetModuleEvent` occurrences to registered listener applications as appropriate for changes in the configuration.

When an HNIMP receives a `ssdp:byebye` message and the device sending the message is in the current device list of the HNIMP, the HNIMP SHALL generate `DeviceEvent` and `NetModuleEvent` occurrences to registered listener applications as appropriate for changes in the configuration. If the device that sent the `byebye` message is not in the current device list of the HNIMP, then the HNIMP SHALL NOT generate any events for that device.

6.2.4 NetActionEvent class

6.2.4.1 `getActionStatus` method

The `getActionStatus` method SHALL be mapped to UPnP action status as per the following table:

Table 6–1 - TransferStatus Return Values

TransferStatus value	Value returned by <code>getActionStatus</code>
STOPPED	ACTION_CANCELLED
ERROR	ACTION_FAILED
COMPLETED	ACTION_COMPLETED

6.2.4.2 `getError` method

When the `getActionStatus` method returns `ACTION_FAILED`, the `getError` method returns UPnP Error code provided by the action response associated with this `NetActionEvent` as defined in [UPNP DA]. This method returns -1 if this action has not failed or has not completed.

6.2.4.3 `getResponse` method

Returns an object derived from a UPnP response. The object is specific to the request; see methods that take a `NetActionHandler` parameter. When the object returned is a `ContentList`, each object type contained is determined by the properties returned in the action response for the object. When a returned UPnP object contains a `upnp:srsRecordScheduleID` property but no `upnp:srsRecordTaskID` property, the HNIMP SHALL create a `NetRecordingEntry` and contain it in the `ContentList`. When a returned UPnP object contains a `upnp:srsRecordTaskID` property, the HNIMP SHALL create a `RecordingContentItem` and contain it in the `ContentList`. Otherwise, the HNIMP SHALL create a `ContentItem` and contain it in the `ContentList`.

6.2.5 NetActionRequest

6.2.5.1 `cancel` method

When the in progress action is a `CDS ImportResource` or `ExportResource` action, the `cancel` method SHALL use the [UPNP CDS] `StopTransferResource` action. If the action is not an in-progress `CDS ImportResource` or `ExportResource` action, then this method SHALL return `false`.

6.2.5.2 `getActionStatus` method

The `getActionStatus` method returns one of the constants defined in the `NetActionEvent` interface. Implementations SHALL map UPnP action Result values to the constants as appropriate.

NOTE (informative): Applications checking the values MAY receive any defined constant value for any action response depending on the implementation-specific mappings of response values to constant values.

6.2.5.3 *getProgress method*

When the action is a CDS ImportResource action, the getProgress method SHALL use the [UPNP CDS] GetTransferProgress action. In this case, when the TransferStatus result value is IN_PROGRESS the value returned SHALL be determined by the following statement:

```

If the GetTransferProgress action TransferTotal or TransferLength arguments are
indeterminate, or if any UPnP errors are encountered
    Return -1.0
Else If (TransferTotal - TransferLength) > 0
    Return ((TransferTotal - TransferLength) / TransferTotal) rounded to one decimal
    place
Else
    Return 1.0

```

If the action is ImportResource and the TransferStatus result value is COMPLETED, this method SHALL return 1.0. If the TransferStatus result value is STOPPED or ERROR, this method SHALL return 0.0. If the action is not ImportResource, this method SHALL return -1.0.

6.2.6 HomeNetPermission class

No UPnP mapping is defined for the HomeNetPermission class.

6.2.7 NetManager class

6.2.7.1 *getDevice method*

The HNIMP SHALL search for the device by comparing the name parameter to the <friendlyName> element in the device description; see [UPNP DA].

6.2.7.2 *getDeviceList method*

The getDeviceList SHALL return a list of org.ocap.hn.Device objects created during discovery as defined in Section 6.1. For each device the list SHALL be ordered by root device, then sub-devices within the root. Local devices SHALL appear in the list first.

6.2.7.3 *updateDeviceList method*

The updateDeviceList method SHALL cause the HNIMP to send out an ssdp:all broadcast discover request as defined in [UPNP DA]. Any changes to home network configuration SHALL generate appropriate DeviceEvent and NetModuleEvent generation.

6.2.8 NetModuleEvent class

No UPnP mapping is defined for the NetModuleEvent class.

6.2.9 NetModule interface

6.2.9.1 *getNetModuleId method*

The getNetModuleId method SHALL return the same value as getProperty when passed the PROP_NET_MODULE_ID constant.

6.2.9.2 *getNetModuleType method*

The getNetModuleType method SHALL return one of the CONTENT constants in the NetModule interface. The constant returned SHALL be determined based on the following criteria:

- CONTENT_SERVER when the NetModule is a ContentServerNetModule.
- CONTENT_RECORDER returned when the NetModule is a RecordingNetModule.

6.2.9.3 *getProperty method*

The `getProperty` method takes a property name string constant defined in the `NetModule` interface and returns a value mapped to device discovery. Values returned from elements in a `<service>` element the `NetModule` was created for and defined by the description defined in [UPNP DA] section 2.1 SHALL be populated as follows:

- PROP_NETMODULE_ID parameter value returned from the `<serviceId>` element.
- PROP_DESCRIPTION_URL parameter value returned from `<SCPDURL>` element.
- PROP_CONTROL_URL parameter value returned from `<controlURL>` element.
- PROP_EventSub_URL parameter value returned from `<eventSubURL>` element.
- PROP_NETMODULE_TYPE parameter value returned from `<serviceType>` element.

6.2.10 PropertyFilter class

No UPnP mapping is defined for the `PropertyFilter` class.

6.3 Package `org.ocap.hn.content`

6.3.1 AudioResource Interface

6.3.1.1 *getSampleFrequency method*

The `getSampleFrequency` method maps to the [DIDL-L] property `res@sampleFrequency`, which is referenced in [UPNP CDS]. This is the sample frequency in Hertz (Hz).

6.3.1.2 *getNumberOfChannels method*

The `getNumberOfChannels` method maps to the [DIDL-L] property `res@nrAudioChannels`, which is referenced in [UPNP CDS]. This integer refers to the number of audio channels within the resource, such as 1 for monaural, 2 for stereo, or 6 for Dolby surround sound.

6.3.1.3 *getBitsPerSample method*

The `getBitsPerSample` method maps to the [DIDL-L] property `res@bitsPerSample`, which is referenced in [UPNP CDS]. This integer refers to the number of bits per sample within the resource.

6.3.1.4 *getLanguages method*

The `getLanguages` method maps to the [DIDL-L] property `dc:language`, which is referenced in [UPNP CDS]. This method SHALL return an array of Strings with one entry for each value associated with the multi-valued `dc:language` property. If the optional `dc:language` property is not present, this method SHALL return a zero length array of Strings.

6.3.2 ContentContainer

A local `ContentContainer` represents CDS entries that are exposed by an OC-DMS on the home network. Cable services such as broadcast services and recordings made from broadcast services SHALL NOT be exposed in a

CDS service browse or search action response unless they were added to a local ContentContainer using either the addContentEntry or addContentEntries methods.

Whenever a ContentContainer object is created, it is associated with a UPnP ContainerID that SHALL be unique across all ContentContainer objects.

6.3.2.1 addContentEntry method

If the ContentContainer on which this method called does not have the parentID property, that is, the ContentContainer is not added to CDS, the HNIMP SHALL throw IllegalStateException.

Otherwise, if the ContentEntry parameter has a parentID property, the HNIMP SHALL remove the ContentEntry parameter from the ContentContainer whose ObjectID is the ID contained in the parentID property, and SHALL add the ContentEntry parameter to this ContentContainer. In this case, the HNIMP changes only the parentID property of the ContentEntry parameter to contain the ObjectID of this ContentContainer, and does not change any other properties of the ContentEntry parameter.

Otherwise, the ContentContainer.addContentEntry method adds the ContentEntry parameter to the ContentContainer. In this case, the process of this method is based on the ContentEntry type or derived ContentEntry type of the ContentEntry parameter. Any properties added to a CDS entry by this method are also added to the ContentEntry parameter.

When the addContentEntry method is called with a parameter Object that implements the ContentEntry interface, but does not implement any other interface that derives from ContentEntry, then the HNIMP SHALL create a CDS item for the ContentEntry. See Section 6.3.3 for a mapping to ContentItem attributes to CDS properties.

When the parameter is a ContentContainer, the HNIMP SHALL create a CDS container and add it to CDS at a location where this ContentContainer is the parent. When the parameter also extends the MediaStorageVolume interface defined in [OCAP DVR], the HNIMP SHALL add the ocap:mediaStorageVolume property and set the value to the Object ID of the container.

When the parameter is a RecordingContentItem, the HNIMP SHALL create a CDS content item and add it to the CDS container corresponding to this ContentContainer. The HNIMP SHALL add the OCAP properties defined in Annex C.1.1 to the CDS content item. The property values SHALL be taken from the Object that implements the RecordingContentItem and OcapRecordingRequest interfaces using the corresponding attributes in the OcapRecordingRequest. If the HNIMP supports SRS, it SHALL perform the following tasks:

1. If the RecordingContentItem does not already contain a upnp:srsRecordScheduleID property, the HNIMP SHALL create a SRS object.recordSchedule.direct.manual as defined by [UPNP SRS]. The HNIMP SHALL add a upnp:srsRecordScheduleID property containing the ID of the RecordSchedule to the CDS content item corresponding to the RecordingContentItem.
2. If the RecordingContentItem does not already contain a upnp:srsRecordTaskID property, the HNIMP SHALL create a SRS object.recordTask as defined by [UPNP SRS] and add the RecordTask to the SRS RecordSchedule whose ID is indicated by the upnp:srsRecordScheduleID property of the RecordingContentItem. The HNIMP SHALL add a upnp:srsRecordTaskID property containing the ID of the RecordTask to the CDS content item corresponding to the RecordingContentItem. The HNIMP SHALL also add ocap:scheduledCDSentryID defined in C.1.2 to the RecordTask.
3. If step 2 described above was performed and the RecordingContentItem is included in a NetRecordingEntry, the HNIMP SHALL add ocap:netRecordingEntry property to the CDS content item corresponding to the RecordingContentItem. The ocap:netRecordingEntry must contain the CDS ObjectID of the NetRecordingEntry. In addition, the HNIMP SHALL add the CDS ObjectID of the RecordingContentItem to a list contained in the ocap:RCIList property of the CDS content item

corresponding to the NetRecordingEntry. The definitions of ocap:netRecordingEntry property and ocap:RCIList property are described in C.1.3.

NOTE: The ocap:RCIList property includes Comma Separated Value (CSV) list which represents the list of the ObjectIDs of the RecordingContentItems added both to the NetRecordingEntry and to the CDS. When the OC-DMS responds to CDS:Browse or CDS:Search, which retrieves the CDS content item corresponding to a NetRecordingEntry, the OC-DMS HNIMP SHALL include the ocap:RCIList property to the <item> block corresponding to the NetRecordingEntry by using ocap namespace <desc> block.

When the parameter is a NetRecordingEntry, the HNIMP SHALL create a CDS content item and add it to the CDS container corresponding to this ContentContainer. If the HNIMP supports SRS, it SHALL create a SRS object.recordSchedule.direct.manual as defined by [UPNP SRS] The HNIMP SHALL add a upnp:srsRecordScheduleID property containing the ID of the RecordSchedule to the CDS content item corresponding to the NetRecordingEntry. The HNIMP SHALL also add an ocap:scheduledCDSEntryID and an ocap:cdsReference to the RecordSchedule. The definitions of ocap:scheduledCDSEntryID and ocap:cdsReference are described in C.1.2. The ocap:scheduledCDSEntryID SHALL contain the object ID of the CDS entry created for the NetRecordingEntry. The ocap:cdsReference SHALL contain the properly escaped DIDL-Lite document of the CDS content item corresponding to the NetRecordingEntry.

Regardless of the type of the ContentEntry parameter, when the created CDS entry corresponding to the ContentEntry parameter is added to CDS, the HNIMP SHALL add a parentID property containing the ObjectID of this ContentContainer to the CDS entry.

Note that this method can only add a local ContentEntry object to a local ContentContainer. If either this ContentContainer or the ContentEntry parameter is not a local object, this method returns false.

6.3.2.2 addContentEntries method

The addContentEntries method causes the same mapping as the addContentEntry method with the addition that the mapping SHALL be made for each ContentEntry passed to this method in the parameter array. When this method is used to add CDS entries, the HNIMP SHALL add all of the parameter content entries before generating corresponding CDS changed events.

6.3.2.3 createContentContainer method

When creating the new content container, the HNIMP SHALL add CDS properties as required by this specification for a CDS container. The container dc:title property SHALL be populated with the title parameter. The HNIMP SHALL populate the ocap:accessPermissions property with the permissions parameter.

6.3.2.4 createContentItem method

When creating the new content item, the HNIMP SHALL create a ContentItem object that maps to a UPnP object of the item class. The item dc:title property SHALL be populated with the title parameter. The HNIMP SHALL populate the ocap:accessPermissions property with the permissions parameter. The HNIMP SHALL create remaining properties for the item as required by [DLNA vol 1]. The HNIMP SHALL create <res> elements for the content parameter as defined by [DLNA vol 1] for a file resource of unknown MIME type.

6.3.2.5 getContainerClass method

The getContainerClass method returns a constant that maps to the UPnP class of the container. See the ContentContainer interface for class constants.

6.3.2.6 *removeContentEntry method*

When the `removeContentEntry` method is called on a local `ContentContainer` and the `ContentEntry` parameter passed to this method is contained in the `ContentContainer`, the HNIMP SHALL perform the following process:

1. When the parameter is a `ContentContainer`, the HNIMP SHALL remove all of the child `ContentEntry` objects of the `ContentContainer` parameter first by calling this method recursively. After that, the HNIMP removes the `ContentContainer` parameter from this `ContentContainer` object.
2. When the parameter is a `RecordingContentItem` that contains a `upnp:srsRecordTaskID` property, the HNIMP removes the SRS object.`recordTask` indicated by the property. The HNIMP also removes the property from the parameter.
3. When the parameter is a `RecordingContentItem` that contains a `upnp:srsRecordScheduleID` property and is not added to any `NetRecordingEntry`, the HNIMP removes the SRS object.`recordSchedule.direct.manual` indicated by the property. The HNIMP also removes the property from the parameter.
4. When the parameter is a `RecordingContentItem` that contains a `upnp:srsRecordScheduleID` property and is added to a `NetRecordingEntry`, the HNIMP removes `ocap:netRecordingEntry` property from the `RecordingContentItem`. In addition, the HNIMP removes the `ObjectID` of the parameter from the `ocap:RCIList` of the `NetRecordingEntry`. The HNIMP also removes the `ObjectID` of the parameter from the `ocap:RCIList` of the CDS content item corresponding to the `NetRecordingEntry`.
5. When the parameter is a `NetRecordingEntry` that contains a `upnp:srsRecordScheduleID` and contains no `RecordingContentItem` objects, the HNIMP removes the SRS object.`recordSchedule.direct.manual` object indicated by the property. The HNIMP also removes the property from the parameter.
6. When the parameter is a `NetRecordingEntry` that contains a `upnp:srsRecordScheduleID` and contains one or more `RecordingContentItems`, the HNIMP throws `IllegalArgumentException`.
7. If no exception was thrown, the HNIMP removes the `ContentEntry` parameter from the `ContentContainer`. The CDS entry corresponding to the `ContentEntry` parameter is also taken from CDS. In addition, the HNIMP removes the `parentID` property from the `ContentEntry` parameter.

If the `removeContentEntry` method is called on a remote `ContentContainer` or the `ContentEntry` parameter is not contained in the `ContentContainer`, the HNIMP SHALL return false.

NOTE (informative): The lifetime of the `RecordTask` after the recording finishes is compliant with the definition in [UPNP SRS].

6.3.2.7 *removeContentEntries method*

The `removeContentEntries` method causes the same mapping as the `removeContentEntry` method with the addition that the mapping SHALL be made for each `ContentEntry` passed to this method in the parameter array. When this method is used to remove CDS entries, the HNIMP SHALL remove all of the parameter content entries before generating corresponding CDS changed events.

6.3.3 **ContentEntry Interface**

For each `ContentEntry`, the implementation SHALL add properties required by [DLNA vol 1] to the metadata entries in the `MetadataNode` contained in the `ContentEntry`.

6.3.3.1 *getID method*

[UPNP CDS] does not export the file name of a resource. The CDS Object ID SHALL be returned by the `getID` method. This is the name of the content entry, and is not the title of a piece of content, but rather an identifier of a physical piece of content. The HNIMP SHALL set the CDS ObjectID for locally hosted ContentEntries.

6.3.3.2 *getContentSize method*

The `getContentSize` method maps to the [DIDL-L] property `res@size`, which is referenced in [UPNP CDS]. This unsigned long value is the size in bytes of the resource. The HNIMP SHALL NOT set the `res@size` property for locally hosted ContentEntries unless directed to do so by an application using the `MetadataNode.addMetadata()` method.

6.3.3.3 *getCreationDate method*

The `getCreationDate` method maps to the [DC] property `dc:date` if available. The HNIMP SHALL NOT set the `dc:date` property for locally hosted ContentEntries unless directed to do so by an application using the `MetadataNode.addMetadata()` method.

6.3.3.4 *getExtendedFileAccessPermissions method*

The `getExtendedFileAccessPermissions` method returns an `ExtendedFileAccessPermission` that maps to the `ocap:accessPermissions` property contained in a CDS entry when available. If that property doesn't exist, the implementation SHALL return null. The HNIMP SHALL NOT set the `ocap:accessPermissions` property for locally hosted ContentEntries that are not `RecordingContentItems` unless directed to do so by an application using the `MetadataNode.addMetadata()` method.

6.3.3.5 *getLocation method*

[UPNP CDS] does not offer file location, so this string can be created using the identifiers of the content and its parent.

6.3.3.6 *getEntryParent method*

If the CDS entry for the parent of the ContentEntry is in cache, the HNIMP SHALL map it to a `ContentContainer` as defined in Section 6.3.2.3 and returned from this method.

6.3.3.7 *getParentID method*

The HNIMP SHALL return the `res@parentID` property of this entry. The HNIMP SHALL set the `res@parentID` property for locally hosted ContentEntries.

6.3.3.8 *getContentClass method*

The `getContentClass` method returns a constant that maps to the UPnP class of the content item. See class constants defined in the `ContentItem` interface.

6.3.4 ContentResource class

6.3.4.1 *getContentLocator method*

The `getContentLocator` method returns a `Locator` that maps to the required `ocap:contentURI` property.

6.3.4.2 *getProtocol method*

The `getProtocol` method returns a String that maps to the protocol contained in the `res@protocolInfo` property.

6.3.4.3 *getNetwork method*

The `getNetwork` method returns a String that maps to the network contained in the `res@protocolInfo` property.

6.3.4.4 *getContentFormat*

The `getContentFormat` method returns a String that maps to the MIME type contained in the `res@protocolInfo` property.

6.3.5 IOStatus interface

6.3.5.1 *isInUse method*

This method SHALL consider an asset to be in use if there is an active network transport protocol session to the URI associated with the `<res>` block represented by the `ContentResource` object, and that URI is hosted by the local OCAP device. An "active network transport protocol session" SHALL be considered to be an active HTTP or RTSP session, or a currently valid UPnP AV Connection as identified by a BCM or `ConnectionManager` `ConnectionID`.

6.3.6 MetadataNode class

A `MetadataNode` object has methods for adding and getting metadata. Where those methods take a "key" parameter, the format of the key string is expected to be in the format defined in [UPNP CDS] section 2.2.20. For example, the `getMetadata` method when passed the value "dc:title" will return the title of the content item in the Dublin Core name space when the property is present.

6.3.6.1 *addMetadata method*

When this method is called, the HNIMP SHALL adhere to the following steps in order:

1. Determine the name space of the property to be added or modified. The key parameter can include the OCAP application default name space (i.e., "ocapApp"), a vendor-defined name space, a standardized name space (e.g., "dc", "upnp", "ocap"), or no name space, which implies the OCAP application default. As specified in Section 5.5.2, the HNIMP uses "ocap" namespace for properties defined in Annex C. If the name space is not "ocapApp" or has not been added by a prior call to the `addNameSpace` method and is consequently unknown, this method throws an `IllegalArgumentException`.
2. Determine permissions of the calling application to write the property. Compare the key parameter to existing property names in the pre-determined name space, and if found, verify the calling application has permission to write to the property based on the permissions of the property. If the calling application does not have permission, throw the appropriate exception; see the method description in the [OCAP HN] specification.
3. Process a key match. If the key match was found in the property name space then over-write the property value. If the value type is an instance of `MetadataNode`, all of the properties that map to it are removed from the associated content item and re-added as defined in Section 6.3.6.1.1. The HNIMP SHALL maintain an association of content item properties to `MetadataNode` fields in an implementation-specific fashion for the lifetime of the `MetadataNode` object. The HNIMP SHALL recursively execute these steps for each field in a `MetadataNode` parameter.
4. Process a new key. If no key match is found, the HNIMP SHALL add a new property or properties to the associated CDS content item using the key and value parameters. If the value type is an instance of

MetadataNode, each of the fields are added to the associated CDS content item as defined in Section 6.3.6.1.1. The HNIMP SHALL recursively execute these steps for each field in a MetadataNode parameter.

6.3.6.1.1 Adding Metadata to CDS

Application metadata items are added to a CDS entry using the desc and desc@nameSpace properties as defined in [UPnP CDS]. In order to allow multiple vendor properties to be embedded within a single <desc> property, an OC-DMS HNIMP SHALL validate <desc> properties with the maxOccurs attribute equal to "unbounded". When a <desc> property is created based on a call to this method, the HNIMP SHALL create a <desc> property and add it to the content item as follows:

If the key parameter contains a name space term and it isn't "ocapApp", the HNIMP SHALL create the name space using the name space term from the key parameter and the URI from the URI parameter. Below is an example of the <desc> element where an application is adding the first property to the "myNameSpace" name space:

```
<desc xmlns:myNameSpace="[URI passed to the addNameSpace method]"
nameSpace="[URI passed to the addNameSpace method]" >
```

If the key parameter includes a name space term that is "ocapApp" or does not include a name space term, the HNIMP SHALL create a name space using the format "urn:schemas-opencable-com". Below is an example of the <desc> element where an application is adding a value without a name space term, or the name space term is "ocapApp" in the key parameter, and consequently the OCAP application default name space is used:

```
<desc xmlns:ocapApp="urn:schemas-opencable-com:ocap-application"
nameSpace="urn:schemas-opencable-com:ocap-application">
```

Metadata values with the same name space SHALL be placed in the same <desc> block. The HNIMP SHALL create an element (property) with a name matching the key parameter, not including any name space term, and a value matching the value parameter and encoded as described below, and add it to the <desc> block created for the name space. The value can be an empty string.

When the value parameter is of type java.lang.String, the HNIMP SHALL NOT serialize the object and instead SHALL encode the String value as defined by [DLNA vol 1]. An HNIMP SHALL serialize a java.lang.String object if it is contained in a serializable object.

When the HNIMP serializes an object as described by the javadoc for this method in [OCAP HN], the HNIMP SHALL encode the object in the DIDL-Lite document using Base64 as defined in [RFC 2045]. In addition, the HNIMP SHALL add the ocapSerializedObject attribute to the element tag corresponding to the object being serialized and set the value to "1". For example:

```
<desc id="descriptorX" xmlns:myNameSpace="urn:schemas-opencable-com:<vendor
added name space>" nameSpace="urn:schemas-opencable-com:<vendor added name
space>">
  <myNameSpace:objectX ocapSerializedObject="1">
    <...objectX Base64 encoding...>
  </myNameSpace:objectX>
</desc>
```

The HNIMP SHALL create an XML document that matches metadata nesting when responding to UPnP actions. For instance, if a MetadataNode contains a property with the key "Band-Members" and that property contains another MetadataNode that contains a property with the key "Guitar-Player", then the element for "Band-Members" would contain a "Guitar-Player" element.

6.3.6.1.2 *MetadataNode Transmission*

When a MetadataNode is transmitted on a home network, it is conveyed in a DIDL-Lite document compliant with [UPNP CDS]. An HNIMP SHALL set the id attribute of a <desc> element to an implementation-specific value. The following example demonstrates a metadata document containing fields as described above with the OCAP application default name space:

```
<DIDL-Lite
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/"
  xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/
    http://www.upnp.org/schemas/av/didl-lite-v2-20060531.xsd
    urn:schemas-upnp-org:metadata-1-0/upnp/
    http://www.upnp.org/schemas/av/upnp-v2-20060531.xsd">
  <item id="18" parentID="13" restricted="0">
    <dc:title>Try a little tenderness</dc:title>
    <upnp:class>object.item.audioItem.musicTrack</upnp:class>
    <res protocolInfo="http-get:*:audio/mpeg:*" size="3558000">
      http://168.192.1.1/audio197.mp3
    </res>
    <desc id="descriptorX" xmlns:ocapApp="urn:schemas-opencable-com:ocap-
application" namespace="urn:schemas-opencable-com:ocap-application">
      <ocapApp:Band-Members>
        <ocapApp:Guitar-Player>John Smith</ocapApp:Guitar-Player>
      </ocapApp:Band-Members>
    </desc>
  </item>
</DIDL-Lite>
```

6.3.6.2 *getMetadata method*

For all properties defined by DLNA and this specification, the HNIMP SHALL return property values from the getMetadata method as java.util.String Objects, unless defined otherwise by this specification.

For all vendor-defined properties contained within a desc block with a desc@nameSpace value of "ocapApp", the HNIMP SHALL return property values from the getMetadata method as java.util.String Objects.

For all vendor-defined properties contained within a desc block with the attribute ocapSerializedObject set to "1", the HNIMP SHALL attempt to deserialize the contained data using both the system and application class loaders. Implementations SHALL NOT attempt to deserialize contained data until an application requests such data using this method. Failure to deserialize such data SHALL result in a ClassDefNotFound exception thrown by this method.

6.3.6.3 *addNameSpace method*

When this method is called, the HNIMP SHALL adhere to the follow steps in order:

1. Evaluate the namespace and URI parameters for [UPNP CDS] compliance with a name space in a CDS content item. If either of the parameters is invalid, this method SHALL do nothing and return successfully.
2. Compare the namespace parameter to "ocapApp", and if it matches, this method SHALL do nothing and return successfully.
3. Compare the namespace parameter to the name spaces already added to the MetadataNode via this method. If a match is found, this method SHALL do nothing and return successfully.

4. Compare the URI parameter to the URI terms already added to the MetadataNode via this method and to URI parameters that have been passed to this method and retained but not yet added to the CDS content item. If a match is found, this method SHALL do nothing and return successfully.
5. Retain the namespace and URI parameter values for use with calls to the addMetadata method. Retention of these values is not persistent across HNIMP reboot.

6.3.6.4 *createMetadataNode method*

The HNIMP SHALL NOT use key parameter to the overloaded form of this method when creating XML properties and transmitting the MetadataNode on a home network.

6.3.7 VideoResource Interface

6.3.7.1 *getColorDepth method*

The getColorDepth method maps to the [DIDL-L] property res@colorDepth, which is referenced in [UPNP CDS]. This unsigned integer value is the encoding characteristic of the resource.

6.3.7.2 *getResolution method*

The getResolution method maps to the [DIDL-L] property res@resolution, which is referenced in [UPNP CDS]. This pattern string identifies the XxY resolution of the resource in pixels (typically an image item or video item). String pattern is of the form: [0-9] + x[0-9] + (one or more digits, 'x', followed by one or more digits).

6.3.8 StreamableContentResource interface

6.3.8.1 *getBitRate method*

The getBitRate method maps to the [DIDL-L] property res@bitrate, which is referenced in [UPNP CDS].

6.3.8.2 *getDuration method*

The getDuration method maps to the [DIDL-L] property res@duration, which is referenced in [UPNP CDS].

6.4 Package Org.ocap.hn.content.navigation

6.4.1 ContentDatabaseFilter class

No UPnP mapping is defined for the ContentDatabaseFilter class.

6.4.2 ContentList class

No UPnP mapping is defined for the ContentList class.

6.4.3 DatabaseQuery class

No UPnP mapping is defined for the DatabaseQuery class.

6.4.4 DeviceFilter class

No UPnP mapping is defined for the DeviceFilter class.

6.5 Package org.ocap.hn.profile.upnp

6.5.1 UPnPConstants interface

6.5.1.1 *actor field*

The actor field maps to the UPnP property upnp:actor, which is defined in the UPnP Content Directory Service [UPNP CDS]. This string value is the name of an actor appearing in a video item.

6.5.1.2 *actor_at_role field*

The actor_at_role field maps to the UPnP property actor@role, which is defined in the UPnP Content Directory Service. This string value is the role of the actor in the video item.

6.5.1.3 *album field*

The album field maps to the UPnP property upnp:album, which is defined in the UPnP Content Directory Service. This string value is the title of the album to which the item belongs.

6.5.1.4 *album_art field*

The album_art field maps to the UPnP property upnp:albumArtURI, which is defined in the UPnP Content Directory Service. This URI value is a reference to album art. Values SHALL be properly escaped URIs, as described in [RFC 3986].

6.5.1.5 *artist field*

The artist field maps to the UPnP property upnp:artist, which is defined in the UPnP Content Directory Service. This string value is the name of the artist.

6.5.1.6 *artist_at_role field*

The artist_at_role field maps to the UPnP property upnp:artist@role, which is defined in the UPnP Content Directory Service. This string value is the role of an artist in a work.

6.5.1.7 *artist_discography field*

The artist_discography field maps to the UPnP property upnp:artistDiscographyURI, which is defined in the UPnP Content Directory Service. This URI value is a reference to the artist's discography. Values SHALL be properly escaped URIs, as described in [RFC 3986].

6.5.1.8 *author field*

The author field maps to the UPnP property upnp:author, which is defined in the UPnP Content Directory Service. This string value is the name of an author of a text item.

6.5.1.9 *author_at_role field*

The author_at_role field maps to the UPnP property upnp:author@role, which is defined in the UPnP Content Directory Service. This string value is the role of an author in the work, such as lyrics.

6.5.1.10 *channel_name field*

The `channel_name` field maps to the UPnP property `upnp:channelName`, which is defined in the UPnP Content Directory Service. This string value is used for identification of channels themselves, or information associated with a piece of recorded content.

6.5.1.11 *channel_number field*

The `channel_number` field maps to the UPnP property `upnp:channelNr`, which is defined in the UPnP Content Directory Service. This integer value is used for identification of tuner channels themselves, or information associated with a piece of recorded content.

6.5.1.12 *comments field*

The `comments` field maps to the UPnP property `upnp:userAnnotation`, which is defined in the UPnP Content Directory Service. This string value is a general-purpose tag where a user can annotate an object with some user-specific information.

6.5.1.13 *contributor field*

The `contributor` field maps to the Dublin Core [DC] property `dc:contributor`, which is referenced in the UPnP Content Directory Service. It is recommended in the UPnP Content Directory Service specification that the `contributor` property include the name of the primary content creator or owner (Dublin Core 'creator' property).

6.5.1.14 *creation_date field*

The `creation_date` field maps to the Dublin Core property `dc:date`, which is referenced in the UPnP Content Directory Service. This string identifies the creation date of a piece of content.

6.5.1.15 *creator field*

The `creator` field maps to the Dublin Core property `dc:creator`, which is referenced in the UPnP Content Directory Service. The value of this string is the primary content creator or owner of the object.

6.5.1.16 *description field*

The `description` field maps to the Dublin Core property `dc:description`, which is referenced in the UPnP Content Directory Service. This string contains a brief description of the content item.

6.5.1.17 *director field*

The `director` field maps to the UPnP property `upnp:director`, which is defined in the UPnP Content Directory Service. This string identifies the name of the director of the video content item.

6.5.1.18 *dvd_region_code field*

The `dvd_region_code` field maps to the UPnP property `upnp:DVDRegionCode`, which is defined in the UPnP Content Directory Service. The value of this integer is the region code of the DVD.

6.5.1.19 *genre field*

The `genre` field maps to the UPnP property `upnp:genre`, which is defined in [UPNP CDS]. This string identifies the name of the genre to which an object belongs.

6.5.1.20 id field

The id field maps to the DIDL-Lite property `didl-lite:(object)@id`, which is referenced in the UPnP Content Directory Service. This string contains an identifier for the object. The value of each object ID property SHALL be unique with respect to the UPnP Content Directory and the server hosting this content.

6.5.1.21 language field

The language field maps to the Dublin Core property `dc:language`, which is referenced in the UPnP Content Directory Service. This string identifies the language as defined by [RFC 3066].

6.5.1.22 long_description field

The long_description field maps to the UPnP property `upnp:longDescription`, which is defined in the UPnP Content Directory Service. This string contains a few lines of description of the content item, and is longer than the Dublin Core *description* field.

6.5.1.23 lyrics_ref field

The lyrics_ref field maps to the UPnP property `upnp:lyricsURI`, which is defined in the UPnP Content Directory Service. This URI refers to the lyrics of the song, or the whole album. Values SHALL be properly escaped URIs, as described in [RFC 3986].

6.5.1.24 media_id field

The media_id field maps to the UPnP property `upnp:toc`, which is defined in the UPnP Content Directory Service. The value of this string is the identifier of an audio CD.

6.5.1.25 playlist field

The playlist field maps to the UPnP property `upnp:playlist`, which is defined in the UPnP Content Directory Service. The value of this string is the name of the playlist to which the item belongs.

6.5.1.26 producer field

The producer field maps to the UPnP property `upnp:producer`, which is defined in the UPnP Content Directory Service. The value of this string is the name of the producer, such as the producer of a video or of a CD.

6.5.1.27 publisher field

The publisher field maps to the Dublin Core property `dc:publisher`, which is referenced in the UPnP Content Directory Service. This string identifies the publisher of the content.

6.5.1.28 radio_band field

The radio_band field maps to the UPnP property `upnp:radioBand`, which is defined in the UPnP Content Directory Service. The value of this string is the radio station frequency band. Recommended values are "AM", "FM", "Shortwave", "Internet", or "Satellite".

6.5.1.29 radio_call_sign field

The radio_call_sign field maps to the UPnP property `upnp:radioCallSign`, which is defined in the UPnP Content Directory Service. The value of this string is a radio station call sign, such as "KIFM".

6.5.1.30 radio_station_id field

The `radio_station_id` field maps to the UPnP property `upnp:radioStationID`, which is defined in the UPnP Content Directory Service. The value of this string is some identification of the broadcast frequency of the radio station, such as "107.7".

6.5.1.31 rating field

The `rating` field maps to the UPnP property `upnp:rating`, which is defined in the UPnP Content Directory Service. The value of this string is the rating of the object's resource, for "parental control" filtering purposes, such as "R", "PG-13", and "X".

6.5.1.32 region field

The `region` field maps to the UPnP property `upnp:region`, which is defined in the UPnP Content Directory Service. The value of this string is some identification of the region associated with the source of the object, such as "U.S.", "Latin America", or "Seattle".

6.5.1.33 relation field

The `relation` field maps to the Dublin Core property `dc:relation`, which is referenced in the UPnP Content Directory Service. This URI property is a reference to related resources. Values SHALL be properly escaped URIs as described in [RFC 3986].

6.5.1.34 rights field

The `rights` field maps to the Dublin Core property `dc:rights`, which is referenced in the UPnP Content Directory Service. The value of this string is information about rights held in and over the resource.

6.5.1.35 scheduled_end_time field

The `scheduled_end_time` field maps to the UPnP property `upnp:scheduledEndTime`, which is defined in the UPnP Content Directory Service. The value of this string is date and time, in the "yyyy-mm-ddThh:mm:ss" format defined by [ISO 8601], used to indicate the end time of a scheduled program, intended for use by timers.

6.5.1.36 scheduled_start_time field

The `scheduled_start_time` field maps to the UPnP property `upnp:scheduledStartTime`, which is defined in the UPnP Content Directory Service. The value of this string is date and time, in the "yyyy-mm-ddThh:mm:ss" format defined by [ISO 8601], used to indicate the start time of a scheduled program, intended for use by timers.

6.5.1.37 storage_medium field

The `storage_medium` field maps to the UPnP property `upnp:storageMedium`, which is defined in the UPnP Content Directory Service. The value of this string indicates the type of storage medium used for the content.

6.5.1.38 title field

The `title` field maps to the Dublin Core property `dc:title`, which is referenced in the UPnP Content Directory Service. The value of this string is the name of the object, such as the title of a song, recording, photograph, or book.

6.5.1.39 track field

The track field maps to the UPnP property `upnp:originalTrackNumber`, which is defined in the UPnP Content Directory Service. This integer value indicates the original track number on an audio CD or other medium.

6.6 Package org.ocap.hn.security

6.6.1 NetAuthorizationHandler interface

6.6.1.1 notifyActivityStart method

When the `NetAuthorizationHandler` was registered using the one argument `setAuthorizationHandler` method, or if it was registered with the three-argument `setAuthorizationHandler` method and the `notifyTransportRequests` parameter was set to true, then the HNIMP SHALL call the `notifyActivityStart` method for every HTTP and RTSP message received.

When the `NetAuthorizationHandler` was registered using the three-argument `setAuthorizationHandler` method and the `notifyTransportRequests` parameter was set to false, then the HNIMP SHALL call the `notifyActivityStart` method in accordance with Section 6.8 of [HN-SEC].

When the `notifyActivityStart` method is called, the HNIMP SHALL NOT process the request until this method returns. If the request is denied, the HNIMP SHALL fail the request and return an error code 401 Not Authorized. If the request is granted, the HNIMP SHALL stream the requested content to the client.

6.6.1.2 notifyActivityEnd method

When an application has set a `NetAuthorizationHandler` object by calling the `NetSecurityManager.setAuthorizationHandler` method, the HNIMP SHALL call the `notifyActivityEnd` method in accordance with Section 6.9 of [HN-SEC].

The HNIMP SHALL match activity end occurrences with activity start occurrences and pass the activity identifier to the `notifyActivityEnd` method that was returned from the corresponding call to the `notifyActivityStart` method.

NOTE (informative): To be notified of a UPnP action that terminates a session, an application can register action names, e.g., "CM::ConnectionComplete", using the `setAuthorizationHandler` method, and be notified via the `notifyAction` method.

6.6.1.3 notifyAction method

When an application has set a `NetAuthorizationHandler` object by calling the overloaded `NetSecurityManager.setAuthorizationHandler` method with the `actionNames` parameter and passes in one or more action names to that method, then the HNIMP SHALL call the `notifyAction` method of that object whenever an action matching one of those names is received. For actions that have an HNIMP server connection identifier, the HNIMP SHALL pass that value as the `activityId` parameter to this method; otherwise the HNIMP SHALL pass the value of -1. The HNIMP SHALL NOT process the action until this method returns action accepted or denied. If this method returns denied, the HNIMP SHALL fail the action and return UPnP error code 606 Action Not Authorized.

6.6.2 NetSecurityManager class

6.6.2.1 revokeAuthorization method

When a privileged application calls the `revokeAuthorization` method with an activity identifier parameter that matches in-progress content streaming that was authorized with the same identifier, the HNIMP SHALL stop the stream and send messages to the recipient appropriate for the transport protocol being used.

6.6.2.2 *setAuthorizationHandler overloaded method*

For the overloaded `setAuthorizationHandler` method with the `actionNames` parameter, the parameter is an array of strings where each string is the name of an action that the authorization application is interested in. See Section 6.6.1.3 for further processing of this parameter. The format of each name is as follows:

```
request_type_name = [action_name]
```

```
action_name = upnp_service_abbrev ":" upnp_action_name
```

```
upnp_service_abbrev = "*" | <UPnP service abbreviation, e.g. "CDS", "CM", "SRS", "AVT", "RCS">
```

```
upnp_action_name = "*" | <UPnP action name, e.g. "PrepareForConnection", "UpdateObject">
```

The asterisk "*" is a wildcard and MAY be used to set multiple action names. The following example sets all actions from any CDS service:

```
"CDS:*"
```

The following example sets all actions from any service:

```
"*:*"
```

The names are case-sensitive and always in all capitals. When an unrecognized name is parsed, the HNIMP SHALL ignore it.

UPnP service abbreviations and action names are defined in respective service specifications. If an element of the `actionNames` parameter contains an unknown service abbreviation or action name or violates the format defined in this section, the method SHALL throw `IllegalArgumentException`.

6.6.2.3 *queryTransaction method*

The `actionName` parameter is the name of the action, and the format of the String has the same definition as defined in Section 6.6.2.2.

6.7 Package `org.ocap.hn.service`

6.7.1 RemoteService interface

No UPnP mapping is defined for RemoteService interface.

6.8 Package `org.ocap.hn.recording`

6.8.1 NetRecordingEntry interface

6.8.1.1 *addRecordingContentItem method*

When the `addRecordingContentItem` method is called on a `NetRecordingEntry` object, the HNIMP SHALL perform the following process:

1. If the `RecordingContentItem` already contains a `upnp:srsRecordScheduleID`, throw an `IllegalArgumentException`.
2. If this `NetRecordingEntry` does not contain a `upnp:srsRecordScheduleID`, throw an `IllegalStateException`.

3. If no exception was thrown, add a `upnp:srsRecordScheduleID` property to the parameter `RecordingContentItem` and set the value of the property to the value in the `NetRecordingEntry` `upnp:srsRecordScheduleID` property, and contain the `RecordingContentItem` within this `NetRecordingEntry`.

6.8.1.2 *getConflictingRecordings method*

When the `getConflictingRecordings` method is called on a non-local `NetRecordingEntry` object, the HNIMP SHALL cause an SRS `GetRecordTaskConflicts` action for each `RecordingContentItem` in the parameter. The list of conflicts from each response will be conglomerated in a `ContentList` that SHALL be returned from the `NetActionEvent.getResponse` method in the event generated when all of the conflicts have been discovered.

6.8.1.3 *removeRecordingContentItem method*

When the `RecordingContentItem` parameter was added to this `NetRecordingEntry`, the HNIMP SHALL perform the following process:

1. If the `RecordingContentItem` contains a `upnp:srsRecordTaskID`, throw an `IllegalArgumentException`.
2. If the `RecordingContentItem` does not contain `upnp:srsRecordTaskID`, remove the `RecordingContentItem` from this `NetRecordingEntry`, and remove `upnp:srsRecordScheduleID` from the `RecordingContentItem`.

NOTE (informative): The sequence of removing a series recording on the OC-DMS device is as follows:

1. Remove all `RecordingContentItems`, each of which represents individual recording from CDS, by calling `ContentContainer.removeContentEntry` method or `ContentContainer.removeContentEntries` method. All related SRS `recordTask` objects are removed.
2. Remove the `RecordingContentItems` from the `NetRecordingEntry` corresponding to the series recording by calling `NetRecordingEntry.removeRecordingContentItem` method.
3. Remove the `NetRecordingEntry` from CDS by calling `ContentContainer.removeContentEntry` method or `ContentContainer.removeContentEntries` method.

6.8.1.4 *getRecordingContentItemIDs method*

When the `getRecordingContentItemIDs` method is called, the HNIMP SHALL return the `java.lang.String` object array that contains the `ObjectIDs` of `RecordingContentItems` extracted from the value of the `ocap:RCIList` property of this `NetRecordingEntry`.

NOTE (informative): This method can work both on a local `NetRecordingEntry` and on a remote `NetRecordingEntry`. On the other hand, `getRecordingContentItems` can work on a local `NetRecordingEntry`. When the OC-DMP device retrieves the remote `NetRecordingEntry` by `CDS:Browse` or `CDS:Search` action, the `ocap:RCIList` property is returned in the response. The OC-DMP HNIMP SHALL create the array of the `ObjectIDs` included in the `ocap:RCIList` property. The OC-DMP HNIMP can use these `ObjectIDs` to retrieve the individual recording by using `CDS:Browse` or `CDS:Search` action.

6.8.2 *NetRecordingRequestManager*

6.8.2.1 *setNetRecordingRequestHandler method*

No UPnP mapping is defined for `setNetRecordingRequestHandler` method.

6.8.2.2 *createNetRecordingEntry method*

No UPnP mapping is defined for `createNetRecordingEntry` method.

6.8.3 NetRecordingRequestHandler interface

Once a privileged application has registered itself as a NetRecordingRequestHandler on an HNIMP DMS device, the HNIMP will call the application based on actions caused by other applications running in remote devices.

6.8.3.1 *notifyDisable method*

When an HNIMP receives a SRS DisableRecordSchedule or DisableRecordTask action and the related RecordScheduleID or RecordTaskID corresponds to a recording published in the CDS, the implementation SHALL call the notifyDisable method of the NetRecordingRequestHandler object set by an application. If no NetRecordingRequestHandler is set, or if the notifyDisable method returns false, the implementation SHALL fail the action with error 720, cannot process the request. The Device InetAddress passed to the method SHALL represent an IP address of the device that sent the action. The ContentEntry parameter SHALL represent the individual or series recording published in the CDS.

6.8.3.2 *notifyDelete method*

When an HNIMP receives a CDS DestroyObject action and the related ObjectID corresponds to a recording published in the CDS, the implementation SHALL call the notifyDelete method of the NetRecordingRequestHandler object set by an application. If no NetRecordingRequestHandler is set, or if the notifyDelete method returns false, the implementation SHALL fail the action with error 720, cannot process the request. The InetAddress parameter passed to the method SHALL represent an IP address of the device that sent the action. The ContentEntry parameter SHALL represent the individual or series recording published in the CDS.

6.8.3.3 *notifyDeleteService method*

When an HNIMP receives a SRS DeleteRecordSchedule or DeleteRecordTask action and the related RecordScheduleID or RecordTaskID corresponds to a recording published in the CDS, the implementation SHALL call the notifyDeleteService method of the NetRecordingRequestHandler object set by an application. P SHALL NOT delete SRS or CDS entries corresponding to the ObjectID unless directed to do so by the NetRecordingRequestHandler as the result of a notifyDeleteService method invocation. If no NetRecordingRequestHandler is set, or if the notifyDeleteService method returns false, the implementation SHALL fail the action with error 720, cannot process the request. The InetAddress parameter passed to the method SHALL represent an IP address of the device that sent the action. The ContentEntry parameter SHALL represent the individual or series recording published in the CDS.

6.8.3.4 *notifyPrioritization method*

When an HNIMP receives an OCAP X_PrioritizeRecordings action, it SHALL call the notifyPrioritization method of the NetRecordingRequestHandler object set by an application. If no NetRecordingRequestHandler is set, or if the notifyPrioritization method returns false, the implementation SHALL fail the action with UPnP error 720, cannot process the request. The InetAddress parameter, which represents an IP address of the request source device passed to the method, SHALL be the object corresponding to the device that sent the action.

When the NetRecordingRequestHandler application receives a prioritization request, it will either deny the action by returning false or accept the action by returning true. When true is returned, the implementation SHALL update fields in the CDS and SRS entries corresponding to any changes made to the RecordingRequest(s) by the handler application. Changing recording prioritization MAY change recording states and resources.

6.8.3.5 *notifyReschedule method*

The HNIMP definition extends the [UPNP SRS] and allows a CreateRecordSchedule to be sent with an srs:item element whose id property is not empty. The effect of this is the receiving device SHALL evaluate the id property to determine if a matching schedule has already been published in CDS. The value of the id is the id of a recordSchedule or the id of a recordTask on the device. If a matching schedule is found, the HNIMP SHALL call

the `notifyReschedule` method of the `NetRecordingRequestHandler` object set by an application. If no `NetRecordingRequestHandler` is set, or the `notifyReschedule` method returns false, the implementation SHALL fail the action with UPnP error 720, cannot process the request. The `InetAddress` parameter passed to the method SHALL represent an IP address of the device that sent the action. If the value of the `id` property in the action is the id of a `recordSchedule`, the `ContentEntry` parameter SHALL be the `NetRecordingEntry` object corresponding to the `recordSchedule`. If the value of the `id` property in the action is the id of a `recordTask`, the `ContentEntry` parameter SHALL be the `RecordingContentItem` object corresponding to the `recordTask`. The `NetRecordingEntry` parameter SHALL contain the property and value pairs received with the action and defined in Table 6–2 as defined in Section 6.8.4.7 in this specification.

When the `NetRecordingRequestHandler` application receives a schedule request, it will either deny the action by returning false or accept the action by returning true. When true is returned, the implementation SHALL update fields in the CDS and SRS entries corresponding to any changes made to the `RecordingRequest` by the handler application.

6.8.3.6 *notifySchedule method*

When an HNIMP receives an SRS `CreateRecordSchedule` action, the HNIMP SHALL check whether the action contains all required SRS properties for `object.recordSchedule.direct.manual`. If not, the HNIMP SHALL fail the action with UPnP error 708. When the HNIMP receives the SRS `CreateRecordSchedule` action with an item element whose `id` property is empty, the HNIMP SHALL call the `notifySchedule` method of the `NetRecordingRequestHandler` object set by an application. If no `NetRecordingRequestHandler` is set, or the `notifySchedule` returns false, the implementation SHALL fail the action with UPnP error 720, cannot process the request. When true is returned, a record schedule for the action will have been created during the process of this method; however, the HNIMP SHALL NOT start a recording based on the schedule, as the `NetRecordingRequestHandler` is responsible for that. The `InetAddress` parameter passed to this method SHALL represent an IP address of the device that sent the action. The `NetRecordingEntry` parameter SHALL contain the property and value pairs received with the action and defined in Table 6–2. If the action does not contain `ocap:scheduledStartDateTime`, `ocap:scheduledChannelID`, `ocap:scheduledChannelIDType`, and `ocap:scheduledDuration`, the HNIMP SHALL create the missing properties among them by using SRS properties and their values in the action, and SHALL add the created properties to the `NetRecordingEntry`. The relationship between these OCAP properties and SRS properties is as represented in Table 6–2. The HNIMP SHALL convert `srs:title` in the action to `dc:title` in the `NetRecordingEntry` parameter. In addition, the HNIMP SHALL add all the property and value pairs received with the action, whose name space is not SRS namespace, to the `NetRecordingEntry` parameter. Note that the HNIMP SHALL NOT add SRS properties to the `NetRecordingEntry` parameter.

When the `NetRecordingRequestHandler` application receives a schedule request, it will either deny the action by returning false or accept the action by returning true. When true is returned, the implementation SHALL verify the `NetRecordingEntry` parameter includes a parent container `Id` property, in order to verify the `NetRecordingRequestHandler` application added a CDS entry for the parameter. If the `NetRecordingEntry` parameter was added to CDS, the HNIMP SHALL succeed this action and return the UPnP response. The HNIMP SHALL contain all the "ocap" and "ocapApp" name space properties received with the action to the response. In this case, the HNIMP creates an association between the `NetRecordingEntry` parameter and the `CreateRecordSchedule` action during the processing of this method as defined in Section 6.3.2.1 of this specification, regarding `NetRecordingEntry`. The HNIMP SHALL update content of the `ocap:cdsReference` in the `RecordSchedule` associated with the `NetRecordingEntry`, if necessary. If the `NetRecordingEntry` was not added to CDS, the HNIMP SHALL fail the action with UPnP error 720, cannot process the request.

Section 6.8.1.1 defines implementation behavior when the `NetRequestHandler` application calls the `NetRecordingEntry.addRecordingContentItem` method using the `NetRecordingEntry` passed to this method.

6.8.3.7 *Event Generation method*

Use of a number of methods in this interface may cause asynchronous event generation.

6.8.3.7.1 *Recording Request Events*

An Object that implements the RecordingContentItem interface SHALL also implement the org.ocap.dvr.OcapRecordingRequest interface. Hence, an SRS RecordTask associated with a RecordingContentItem as defined by Section 6.8.3.6 is also associated with the corresponding OcapRecordingRequest. When the implementation generates events for a RecordTask based on RecordingRequest state changes, the implementation SHALL map the RecordingRequest to RecordTask as shown in Table C-2. Whenever CDS properties in the OCAP name space change, the HNIMP SHALL generate a CDS changed event.

6.8.4 **RecordingNetModule interface**

6.8.4.1 *requestDelete method*

The requestDelete method SHALL use the [UPNP CDS] DestroyObject action for the ObjectID of the CDS entry associated with the RecordingContentItem parameter.

6.8.4.2 *requestDeleteService method*

If the ContentEntry passed to this method is a RecordingContentItem, the requestDeleteService method SHALL use the DeleteRecordTask action for the ObjectID included in the upnp:srsRecordTaskID contained in the ContentEntry parameter. If the ContentEntry passed to this method is a NetRecordingEntry, the requestDeleteService method SHALL use the DeleteRecordSchedule action for the ObjectID in the upnp:srsRecordScheduleID contained in the ContentEntry parameter.

6.8.4.3 *requestDisable method*

The requestDisable method SHALL use the [UPNP SRS] DisableRecordSchedule action when the parameter is a NetRecordingEntry, or the DisableRecordTask action when the parameter is a RecordingContentItem. The implementation SHALL use the RecordScheduleID and RecordTaskID, respectively, associated with the parameter.

6.8.4.4 *getRootContainer method*

The root container for the module is returned during discovery, and no mapping for this method is needed.

6.8.4.5 *requestPrioritize methods*

The requestPrioritize methods SHALL use the OCAP X_PrioritizeRecordings action defined in Annex C.2.1. The parameter SHALL be converted to a list of RecordSchedule IDs or RecordTask IDs, depending on whether an array of NetRecordingEntry objects or an array of RecordingContentItem object is passed in, respectively; see the javadoc for both variants of this method.

6.8.4.6 *requestReschedule method*

The requestReschedule method maps to UPnP actions the same way as the requestSchedule method, except the HNIMP SHALL populate the srs:item element of the [UPNP SRS] CreateRecordSchedule action with the srs:@id property of the srs:item of a schedule that was created in a previous call to the requestSchedule method. If the ContentEntry parameter passed to this method is the NetRecordingEntry object, the HNIMP SHALL set the value of upnp:srsRecordScheduleID property of the NetRecordingEntry object to the srs:@id property in the action. If the ContentEntry parameter is the RecordingContentItem object, the HNIMP SHALL set the value of upnp:srsRecordTaskID property of the RecordingContentItem object to the srs:@id property in the action.

If the ContentEntry parameter is neither the NetRecordingEntry with upnp:srsRecordScheduleID property nor the RecordingContentItem with upnp:srsRecordTaskID property, or if the NetRecordingSpec parameter does not contain the required properties for recording, the HNIMP SHALL throw the IllegalArgumentException.

6.8.4.7 requestSchedule method

The requestSchedule method SHALL use the [UPNP SRS] CreateRecordSchedule action and populate the messages so that the receiving device can create an object.recordschedule.direct.manual class as defined by [UPNP SRS] section 2.9.3.1.1. The calling application is responsible for adding metadata to the NetRecordingSpec parameter with all required properties except for srs:@id and srs:class for creating an object.recordschedule.direct.manual class on a remote device. If the NetRecordingSpec parameter does not include all required properties except for srs:@id and srs:class, the HNIMP SHALL throw IllegalArgumentException. Application-specific metadata is added to the CreateRecordSchedule action using the <desc> element as described in Section 6.3.6.1.2. The [UPNP SRS] specification table C.7 defines required and optional properties applicable to the object.recordschedule.direct.manual class. In addition, Annex C.1.1 defines properties for the OCAP name space, and the HNIMP SHALL support OCAP properties for purposes of remote recording scheduling in a NetRecordingSpec as specified the left column in the table below. If the ocap:scheduledStartDateTime is included in the NetRecordingSpec parameter, the implementation SHALL copy its value to srs:scheduledStartDateTime property in the CreateRecordSchedule action and SHALL NOT add the ocap:scheduledStartDateTime property. The same rule SHALL be applied to ocap:scheduledChannelID and srs:scheduledChannelID pair, ocap:scheduledChannelID@type and srs:scheduledChannelID@type pair, and ocap:scheduledDuration and srs:scheduledDuration pair. The implementation SHALL ignore srs:@id and srs:class in the NetRecordingSpec parameter if they are specified. The implementation SHALL add the application-specific metadata in the NetRecordingSpec to the CreateRecordSchedule action using DIDL-Lite <desc>block as defined in 6.3.6.1.2 of this specification. Note that the default namespace of the XML document of CreateRecordSchedule action is srs: namespace (whose namespace URI is "urn:schema-upnp-org:av:srs") and that <desc> element is defined in didl-lite namespace (whose namespace URI is "urn:schema-upnp-org:metadata-1-0/DIDL-Lite/"). Therefore, unlike adding <desc> block to CDS, the implementation SHALL add the namespace declaration of didl-lite namespace to the document as defined in XML namespace specification [XML NS]. In addition, the HNIMP SHALL add <didl-lite:DIDL-Lite> element to the document and add <didl-lite:desc> elements inside the <didl-lite:DIDL-Lite> element because the <didl-lite:desc> element cannot be a direct child element of the srs:item element as defined by DIDL-Lite XML schema.

NOTE (informative): The calling application should not add optional SRS properties to the NetRecordingSpec parameter.

Table 6-2 - Recording Properties

OCAP Property	Use for SRS property if specified
ocap:scheduledStartDateTime	srs:scheduledStartDateTime
ocap:scheduledChannelID	srs:scheduledChannelID
ocap:scheduledChannelIDType	srs:scheduledChannelID@type
ocap:scheduledDuration	srs:scheduledDuration
ocap:priorityFlag	
ocap:retentionPriority	
ocap:accessPermissions	
ocap:organization	
ocap:appID	
ocap:msoContentIndicator	
ocap:expirationPeriod	

The calling application MAY add additional properties in the "ocapApp" namespace as defined in Section 6.8.5. The HNIMP SHALL include any properties in the NetRecordingSpec in the corresponding SRS:CreateRecordSchedule action.

When the action returns successfully, the action response includes ocap:scheduledCDSentryID and ocap:cdsReference. Then, the implementation SHALL create a NetRecordingEntry for the scheduled recording from the ocap:cdsReference property value in the action response, which includes all of the properties of the CDS item corresponding to the recordSchedule in the action response. The implementation SHALL associate the CreateRecordSchedule action response RecordScheduleID, Result, and UpdateId with the NetRecordingEntry created on the DMP. See Appendix I.1 for the sequence of events caused by the CreateRecordSchedule action in an HNIMP device. The NetRecordingEntry SHALL be returned from the NetActionEvent.getResponse method from the NetActionEvent created for requestSchedule response.

6.8.5 RecordingContentItem interface

When a RecordingContentItem is created as a result of org.ocap.dvr.shared.RecordingRequest creation, the HNIMP SHALL set the upnp:class property to "object.item.videoItem" and populate the properties in the RecordingContentItem using corresponding OCAP properties and fields from the corresponding org.ocap.dvr.shared.RecordingRequest. See section C.1.1 for OCAP property definitions. When a RecordingContentItem is created by any means, the HNIMP SHALL include a TSpec as defined in Annex D. When the HNIMP creates a RecordedService for the RecordingRequest, it SHALL add a <res> element to the CDS entry for the RecordingContentItem that corresponds to the recorded content. If a RecordingRequest is deleted, HNIMP SHALL delete the corresponding RecordingContentItem, any associated metadata, and any SRS and CDS entries.

6.8.5.1 requestConflictingRecordings method

The requestConflictingRecordings method SHALL map to the [UPNP SRS] GetRecordTaskConflicts action. The HNIMP SHALL use the RecordTaskID determined from the requestSchedule response for this RecordingContentItem; see [UPNP SRS] RecordTaskID definition. When the HNIMP receives a successful response for the GetRecordTaskConflicts action, it SHALL send a CDS Search action and retrieve the CDS items by searching for upnp:srsRecordTaskID property values that coincide with the GetRecordTaskConflicts results.

When successful, the ContentList result SHALL be returned via the corresponding NetActionEvent.getResponse method. Each entry in the list SHALL be a RecordingContentItem returned from the CDS search.

When either the GetRecordTaskConflict or the Search action is unsuccessful, a NetActionEvent is generated and no other UPnP action traffic for this method is instigated.

6.8.5.2 requestSetMediaTime method

The requestSetMediaTime method maps to the UpdateObject action; see [UPNP CDS]. The client and server HNIMP SHALL use the parameter as the new value for the ocap:mediaPresentationPoint property in the CDS content item corresponding to the RecordingContentItem. The server HNIMP SHALL update the RecordedService associated with the RecordingContentItem with the new value of the ocap:mediaPresentationPoint property, with the same effect as calling setMediaTime() on that RecordedService.

6.8.5.3 getRecordingEntryID method

The getRecordingEntryID method returns the ObjectID that is contained in ocap:netRecordingEntry property of this RecordingContentItem. If the ocap:netRecordingEntry property was not added to this RecordingContentItem, this method SHALL return null.

NOTE (informative): A RecordingContentItem has an ocap:netRecordingEntry only when the RecordingContentItem is added both to a NetRecordingEntry and to CDS.

6.8.5.4 *ocap:mediaPresentationPoint* property

The server HNIMP SHALL set the *ocap:mediaPresentationPoint* property to the value of the media time that is set by the method `org.ocap.shared.dvr.RecordedService.setMediaTime` or the first media time of the `RecordedService` if `setMediaTime` has never been called on this `RecordedService`.

The client HNIMP SHALL start playback of a `RecordingContentItem` from the media time that is specified in the *ocap:mediaPresentationPoint* property. For example, if HTTP is being used, the client HNIMP MAY use the `TimeSeekRange.dlna.org` header.

Following any changes to the *ocap:mediaPresentationPoint* property in a server `RecordingContentItem`, the server HNIMP SHALL update the `RecordedService` associated with the `RecordingContentItem` with the new value of the *ocap:mediaPresentationPoint* property, with the same effect as calling `setMediaTime()` on that `RecordedService`.

6.9 Package `javax.tv.selection`

6.9.1 `ServiceContext` Interface

6.9.1.1 *select* method

The *select* method establishes a means for an application in a DMP to playback a remote recording. When the *select* method is called, the HNIMP SHALL supply `ServiceMediaHandler` objects necessary to render the format type of the selected content. When starting a `ServiceMediaHandler` (Player), the UPnP Control Point SHALL perform the necessary UPnP Connection Manager functions:

- a) Server and Client Negotiations
 - Get capabilities
 - Negotiation of delivery protocol
 - Negotiation of content format / options
- b) Connection Setup
 - Reserve resources
 - Register QoS (if available)
 - Verify access permissions
 - Get AV Transport handle from server (push) or renderer (pull), (optional when supported)

The HNIMP MAY establish Quality of Service for the playback of the remote recording stream using RSD Controller functionality as specified in [RSD PROT] specifications.

Figure 6–1 illustrates `ServiceContext` states and event generation during remote service presentation.

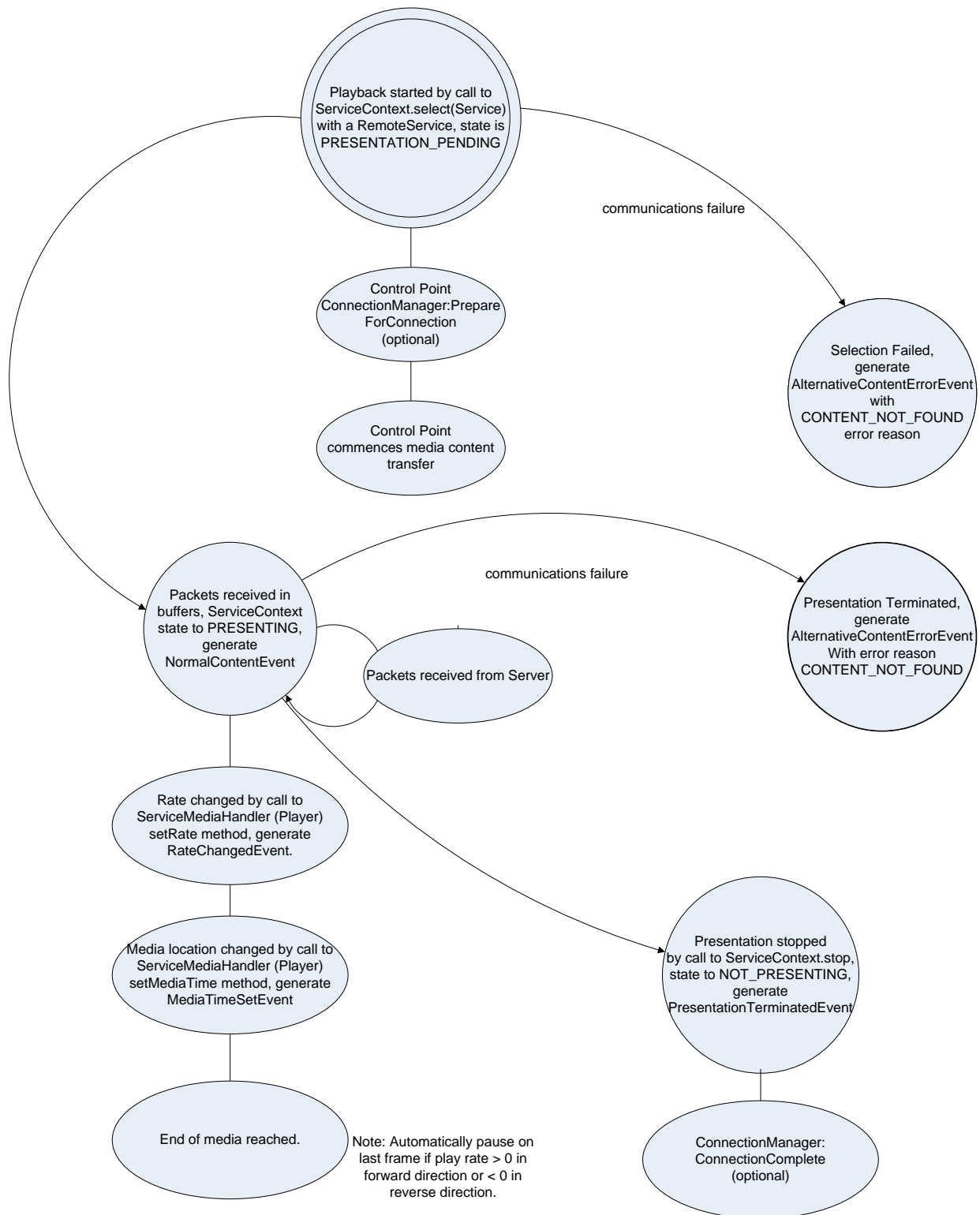


Figure 6–1 - ServiceContext States during RemoteService Presentation

Once the connection has been established, the JMF player MAY provide an interface to connection manager functions (for connection status) if supported by both the OC-DMP and OC-DMS. When HTTP is used, the OC-

DMP SHALL provide playback control (e.g., fast forward, rewind). When the presentation is ended due to failure or consumer selection of a different presentation, the control point implementation is responsible for closing the connection and MAY cause the optional `ConnectionManager::ConnectionComplete` action.

Supported JMF player controls will vary widely from renderer to renderer. A simple audio device, for example, may only support the `GainControl` object and associated methods. More sophisticated AV renderers may support trick-play (`setRate()`). A complete list of supported JMF controls supported by a given module are provided by the player's `getControls()` method.

Once the connection manager has established the connection protocol and prepared the connection, the control point will communicate with the Server to begin media transfer. [DLNA vol 1] allows a number of protocols to be used for media transfer based on client/server negotiations. Annex B describes some techniques that can provide smooth trick-mode presentation using HTTP.

6.9.1.2 *destroy method*

If any connections for remote service presentation are open, the `destroy` method SHALL cause them to be closed based on the protocol used to establish the connections. Any remote resources used for the presentation SHALL be released when the connections are closed. Once connections are closed, further network actions SHALL NOT be taken by subsequent calls to the `destroy` or `stop` methods.

6.9.1.3 *stop method*

The `stop` method SHALL cause the same network actions as the `destroy` method.

6.10 Package `org.javax.media` (JMF)

6.10.1 Clock interface

6.10.1.1 *setMediaTime method*

The `setMediaTime` method SHALL cause a jump to the media time parameter. For network actions, implementations SHOULD handle this in a manner specific to the negotiated transfer protocol. For example, if HTTP is being used the control point MAY use the `TimeSeekRange.dlna.org` header.

6.10.1.2 *setRate method*

The `setRate` method SHALL cause a change in presentation play speed based on the media time parameter. For network actions, implementations SHOULD handle this in a manner specific to the negotiated transfer protocol. For example, if HTTP is being used, the control point MAY use the `PlaySpeed.dlna.org` header.

6.10.1.3 *stop method*

This `stop` method SHALL cause the same network actions as `ServiceContext.stop`; see Section 6.9.1.3.

6.10.2 Controller interface

6.10.2.1 *realize method*

If the media source Server protocol information has not been determined by the control point when the `realize` method is called, the control point SHALL cause a `ConnectionManager::GetProtocolInfo()` action to the source OC-DMS.

6.10.2.2 *prefetch method*

In order to reduce start-up latency, the control point MAY download media content based on the protocol negotiated with the Server.

6.10.2.3 *deallocate method*

The deallocate method SHALL cause the same network actions as ServiceContext.stop; see Section 6.9.1.3.

6.10.2.4 *close method*

The close method SHALL cause the same network actions as ServiceContext.stop; see Section 6.9.1.3.

6.10.3 Player interface**6.10.3.1 *start method***

The start method starts presentation and SHALL initiate media download if not already initiated by the Controller.prefetch method defined in Section 6.10.2.2. The start method SHALL commence or continue ongoing media download for presentation based upon the protocol negotiated with the media content Server.

Annex A XML Documents (Normative)

A.1 OCAP Root Device Description

Note the <deviceType> element SHALL be set to "OCAP-HOST" when the device is an OpenCable Set-top or "OCAP-TERMINAL" when the device is an OpenCable terminal. See [HOST 2.1] for OpenCable Set-top and Terminal definitions.

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>
      urn:schemas-opencable-com:device:OCAP_HOST|OCAP_TERMINAL:1
    </deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>long user-friendly title</modelDescription>
    <modelName>model name</modelName>
    <modelName>model number</modelName>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <ocap:X_MiddlewareProfile xmlns:ocap="urn:schemas-cablelabs-com:device-1-0">OCAP profile
    </ocap:X_MiddlewareProfile>
    <ocap:X_MiddlewareVersion xmlns:ocap="urn:schemas-cablelabs-com:device-1-0">vendor version
    </ocap:X_MiddlewareVersion>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
      XML to declare other icons, if any, go here
    </iconList>
    <deviceList>
      Description of embedded devices added by UPnP vendor
      (if any) go here
    </deviceList>
    <presentationURL>URL for presentation</presentationURL>
  </device>
</root>
```

A.2 OC-DMS Device Description

The OC-DMS device description is based on the UPnP MediaServer XML template.

```
<?xml version="1.0" ?>
```

```

<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
    <deviceType>urn:schemas-upnp-org:device:MediaServer:2</deviceType>
    <ocap:X_OCAPHN xmlns:ocap="urn:schemas-cablelabs-com:device-1-0">OC-DMS-
1.0</ocap:X_OCAPHN>
    <dlna:X_DLNA DOC xmlns:dlna="urn:schemas-dlna-org:device-1-0">
      DMS-1.50
    </dlna:X_DLNA DOC>
    <UDN>uuid:UUID</UDN>
    <friendlyName>friendlyName</friendlyName>
    <manufacturer>manufacturer</manufacturer>
    <manufacturerURL>manufactureURL</manufacturerURL>
    <modelName>modelName</modelName>
    <modelNumber>modelNumber</modelNumber>
    <modelDescription>modelDescription</modelDescription>
    <serialNumber>serialNumber</serialNumber>
    <presentationURL>URL for Presentation</presentationURL>
    <ocap:X_PacingSupported xmlns:ocap="urn:schemas-cablelabs-com">
      Yes or No
    </ocap:X_PacingSupported>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-
org:service:ConnectionManager:2</serviceType>
        <serviceId>urn:upnp-org:serviceId:ConnectionManager</serviceId>
        <SCPDURL>ConnectionManager.xml</SCPDURL>
        <eventSubURL>ConnectionManager/Event</eventSubURL>
        <controlURL>ConnectionManager/Control</controlURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-
org:service:ContentDirectory:3</serviceType>
        <serviceId>urn:upnp-org:serviceId:ContentDirectory</serviceId>
        <SCPDURL>ContentDirectory.xml</SCPDURL>
        <eventSubURL>ContentDirectory/Event</eventSubURL>
        <controlURL>ContentDirectory/Control</controlURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-
org:service:AVTransport:2</serviceType>
        <serviceId>urn:upnp-org:serviceId:AVTransport</serviceId>
        <SCPDURL>AVTransport.xml</SCPDURL>
        <eventSubURL>AVTransport/Event</eventSubURL>
        <controlURL>AVTransport/Control</controlURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-
org:service:ScheduledRecording:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:ScheduledRecording</serviceId>
        <SCPDURL>ScheduledRecording.xml</SCPDURL>
        <eventSubURL>ScheduledRecording.xml/Event</eventSubURL>
        <controlURL>ScheduledRecording.xml/Control</controlURL>
      </service>
    </serviceList>
  </device>
</root>

```

Annex B Remote Playback with Time-Shift

As described in Section 6.9.1.1, when the ServiceContext.select method is called with a Locator that references a remote service, the client and server create a connection and begin normal speed playback. This Annex describes two techniques that allow a client to control playback speed of streaming audio/video. The definitions start at a point after the connection has been setup by a control point.

B.1 HTTP Push Model for MPEG Content

The HTTP Push Model defined in this section provides a technique for video streaming where the server paces the data and the client decodes it as it is received. The HTTP Push Model is instigated by a control point that resides in the client device. To instigate video streaming in the push model, the control point sends an HTTP get message with a ServersidePacedStream header and the URL of the video to stream. When the server does not support the ServersidePacedStream header, it ignores it.

B.1.1 Normal Play

When the server supports the ServersidePacedStream header, it SHALL respond to an HTTP GET message containing the header by pacing blocks of data to the client based on the 27Mhz system clock and MPEG time stamps within the data. It continues to push blocks of data without additional HTTP requests from the client. Each block of data is a response to the initial HTTP GET message that requested the video. The server SHALL setup the blocks of data so that they contain one or more complete groups of pictures or frames as requested by the client and where the PTS SHALL index the first and last I and P frames in the data block at a minimum. The server SHALL NOT send a block of data until the timestamp has matured. Servers MAY take into account network delay when making this determination. The client SHALL decode the data blocks as they are received. The client is responsible for handling de-jitter requirements defined in the [OCAP HN].

B.1.2 Trick Modes

A client can modify the rate at which a server paces the blocks of data using the PlaySpeed.dlna.org header. It is recommended by [DLNA vol 1] that the connection be closed and re-opened when changing the play speed. Server pacing is turned off by default when a new connection is opened and the ServersidePacedStream header is needed in the initial HTTP GET message to move the content output to a different speed. In addition, a TimeSeekRange header will be needed to get to the correct location in the content. When the PlaySpeed.dlna.org header is used, the server scales the data to the maximum bandwidth allowed or lower. At speeds greater than normal play rate, servers MAY send selected I, P, or all frames in order to match the play rate.

B.2 HTTP Pull Model for MPEG Content

The HTTP Pull Model defined in this section provides a technique for video streaming where the client requests data at a rate of its choosing and applies flow control if necessary using TCP-IP. This model is basically the same mechanism provided by DLNA, with the addition of HTTP header definition. The intent of the additional HTTP headers is to provide the ability to smooth out the presentation to some indeterminate extent when both the client and the server support the optional headers. To instigate the pull model, the client sends an HTTP get message with a URL to the video content. Any optional headers defined by this specification SHALL be ignored by servers that do not support them.

B.2.1 Normal Play

In this mode, OC-DMP simply sends HTTP-GET request and receives the requested data. OC-DMP feeds the data to the decoder at the decoder's rate of consumption. Once a connection is established, the client uses HTTP GET messages to request data representing parts of the service. Headers defined by HTTP 1.1, DLNA, and this

specification MAY be used as determined appropriate by the client and as supported by the server. In this mode, it is up to the OC-DMS to use Chunked Transfer Encoding in response to HTTP/1.1 GET request.

B.2.2 Trick Modes

Because of the maximum bandwidth requirement play speeds faster than normal, play SHALL be requested using the PlaySpeed.dlna.org header. It is recommended by [DLNA vol 1] that the connection be closed and re-opened when changing the play speed.

In this mode, at least the following headers are used:

- HTTP GET request
 - PlaySpeed.dlna.org with requested play speed
 - ChunkEncodingMode with "chunk=GOP"
 - MaxGOPsPerChunk with "gops=1"
 - FrameTypeInTrickMode with preferred frame type
 - FrameRateInTrickMode without a value.
- HTTP Response
 - PlaySpeed.dlna.org with selected play speed
 - MaxGOPsPerChunk with "gops=1"
 - FrameTypeInTrickMode with the selected frame type
 - FrameRateInTrickMode with the selected frame rate

The OC-DMP specifies the requested play speed by using the PlaySpeed.dlna.org header in the request. In addition, the OC-DMP MAY request the OC-DMS to use Chunked Transfer Coding, to create each chunk composed of one GOP, to select preferred frame type, and to report the frame rate selected by the OC-DMS.

The OC-DMS that receives such a request decides the actual play speed. OC-DMS then decides the frame rate for this request, considering the current play speed, bandwidth restriction, selected frame type, and so on. The OC-DMS sends the response by using Chunked Transfer Coding, in which each chunk is made up of one GOP. Chunked Transfer Coding makes it easy for the OC-DMP to detect GOP or frame boundaries.

TimeSeekRange.dlna.org can be used by OC-DMP to specify the correct starting point in the content to the OC-DMS.

Annex C OCAP Name Space

Properties, actions, and action arguments are defined in an OCAP XML name space in cases where UPnP does not provide direct mappings for OCAP attributes that must be conveyed in UPnP actions.

C.1 Properties

C.1.1 RecordingContentItem CDS object property definitions

The following properties are mapped from the OCAP DVR RecordingRequest interface to a corresponding RecordingContentItem and CDS videoItem object. When a CDS videoItem representing a RecordingContentItem has been published to the network, these properties are mapped from the local RecordingRequest to the local CDS by the host implementation. Property values in the CDS are implicitly updated by the host as RecordingRequest parameters are updated, whether through normal DVR schedule operation or through manipulation by local OCAP applications. The properties listed in Table 6–2 are also used for requesting the scheduled recording.

Table C–1 - Recording Request Properties Mapped to CDS

Property Name	Name Space	Data Type	Multi-Valued	Reference
taskState	ocap	xsd:int	NO	
scheduledStartDateTime	ocap	xsd:dateTime	NO	
scheduledDuration	ocap	xsd:string	NO	
scheduledChannelID	ocap	xsd:string	NO	
scheduledChannelIDType	ocap	xsd:string	NO	
destination	ocap	xsd:string	NO	
priorityFlag	ocap	xsd:int	NO	
retentionPriority	ocap	xsd:int	NO	
accessPermissions	ocap	xsd:string	NO	
organization	ocap	xsd:string	NO	
appID	ocap	xsd:long	NO	
spaceRequired	ocap	xsd:long	NO	
contentURI	ocap	xsd:string	NO	
mediaPresentationPoint	ocap	xsd:long	NO	
msoContentIndicator	ocap	xsd:boolean	NO	
expirationPeriod	ocap	xsd:long	NO	
mediaStorageVolume	ocap	xsd:boolean	NO	
mediaFirstTime	ocap	xsd:long	NO	

C.1.1.1.1 taskState

Namespace: ocap

Property Data Type: xsd:string

Multi-Valued: YES

Description: The taskState property indicates the current state of this recording for both the SRS RecordTask and the OCAP RecordingRequest. OC-DMS devices SHALL map the values of this property to the state of the local OCAP DVR RecordingRequest state as shown in Table C-2. The property is a CSV string containing first the string form of the RecordingRequest state followed by the srs:taskState property. The string form of the RecordingRequest state SHALL be one of the state names taken from the "OCAP DVR RECORDING STATE" column in Table C-2.

Default Value: N/A – Output Only

Sort Order: Sequenced Numeric

Property Key: "ocap:taskState"

Table C-2 - Recording Request State Mapping

OCAP DVR RECORDING STATE	UPNP SRS RECORDING TASK STATE
PENDING_NO_CONFLICT	IDLE.READY
PENDING_WITH_CONFLICT	IDLE.ATRISK
IN_PROGRESS	ACTIVE.TRANSITION.FROMSTART
IN_PROGRESS_INCOMPLETE	ACTIVE.TRANSITION.RESTART
IN_PROGRESS_WITH_ERROR	ACTIVE.NOTRECORDING
IN_PROGRESS	ACTIVE.RECORD.FROMSTART.OK
IN_PROGRESS_INSUFFICIENT_SPACE	ACTIVE.RECORDING.FROMSTART.ATRISK
IN_PROGRESS_INCOMPLETE	ACTIVE.RECORDING.RESTART.OK
<NO MAPPING>	ACTIVE.RECORDING.RESTART.ATRISK
COMPLETE	DONE.FULL
INCOMPLETE	DONE.PARTIAL
FAILED	DONE.EMPTY
CANCELLED	DONE.EMPTY
TEST	
DELETED	DONE.EMPTY

C.1.1.1.2 Start Time

Namespace: ocap **Property Data Type:** xsd:dateTime **Multi-Valued:** NO

Description: The scheduledStartDateTime property indicates the date and time that the recording is scheduled to start

Default Value: N/A – Output Only

Sort Order: Sequenced date and time

Property Key: "ocap:scheduledStartDateTime"

C.1.1.1.3 Duration

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

Description: The scheduledDuration property indicates the duration requested for recording. This duration does not include any adjustments. These are reflected in the srs:scheduledDurationAdjust property. The syntax of the scheduledDuration property is the same as the srs:scheduledDuration property defined in [UPNP SRS].

Default Value: N/A – Output only.

Sort Order: Property Specific, based on elapsed time. Ascending: shortest elapsed time first.

Property Key: "ocap:scheduledDuration"

C.1.1.1.4 Source (OCAP Locator)

Namespace: ocap **Property Data Type:** xsd:int **Multi-Valued:** NO

Description: The ocap:scheduledChannelID property provides channel information. Its format depends on the associated ocap:scheduledChannelIDType property. The definition of this property is the same as srs:scheduledChannelID property except for using ocap:scheduledChannelIDType property for type property. See [UPNP SRS] Appendix B.4.2. When ocap:scheduledChannelID@type is "SI", the format of the fields is defined by Section 5.5 of this specification.

Default Value: N/A – Output only.

Sort Order: Same as ocap:scheduledChannelIDType property.

Property Key: "ocap:scheduledChannelID"

C.1.1.1.5 Recording Destination

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

Description: The destination property indicates the URI where the recorded content will be saved. It is implementation-specific regarding whether this property matches the ocap:contentURI property.

Default Value: N/A – Output Only

Sort Order: String alpha-numeric

Property Key: ocap:destination

C.1.1.1.6 Recording with Conflicts Priority

Namespace: ocap **Property Data Type:** xsd:int **Multi-Valued:** NO

Description: The priorityFlag property indicates whether a recording should be made even if resource conflicts exist. The value SHALL adhere to the definition in [OCAP DVR].

Default Value: N/A – Output Only

Sort Order: Sequenced Numeric

Property Key: ocap:priorityFlag

C.1.1.1.7 Recording Retention Priority

Namespace: ocap **Property Data Type:** xsd:int **Multi-Valued:** NO

Description: The retentionPriority property indicates the priority of storage retention after a recording has expired. The value SHALL adhere to the definition in [OCAP DVR].

Default Value: N/A – Output Only

Sort Order: Sequenced Numeric

Property Key: ocap:retentionPriority

C.1.1.1.8 Extended File Access Permissions

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

Description: The accessPermissions property indicates the extended file access permissions required to read or write the content. The heterogeneous CSV string SHALL have the following BNF:

```
accessPermissions = world "," read "," write "," owner_org "," read "," write "," owner_app "," read "," write
                    ","
                    [1 * (other_org "," read "," write)]
world = "w="
owner_org = "o=" hex_string
owner_app = "a=" hex_string
other_org = "r=" hex_string
read = "1" | "0"
write = "1" | "0"
hex_string = "0x" 1*hex
hex = digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" | "d" | "e" | "f"
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

Where the terms are given the following definitions:

Term	Definition
world	Read and write access for all applications.
owner_org	Read and write permission for applications in the same organization of the owner application. This value will match the organization_id in the application_identifier signaling found in an AIT or XAIT.
owner_app	Read and write permissions for the application that owns the content. This value will match the application_id in the application_identifier signaling found in an AIT or XAIT.
other_org	Read and write permissions for organizations other than the owning organizations. This is an optional field, and zero to many organizations MAY be listed. This value will match the organization_id in the application_identifier signaling found in an AIT or XAIT.
read	When the value is "1" read permission is given. When the value is "0", read permission is taken away.
write	When the value is "1" write permission is given. When the value is "0", write permission is taken away.

Default Value: N/A – Output Only

Sort Order: N/A multi-valued

Property Key: ocap:accessPermissions

C.1.1.1.9 organization string

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

Description: The organizationString property indicates the organization that this recording is tied to. This field is used to authenticate playback by matching this property to an organization name field in any playback application's certificate chain, as defined in [OCAP DVR].

Default Value: N/A – Output Only

Sort Order: Sequenced Numeric

Property Key: ocap:appID

C.1.1.1.10 AppID

Namespace: ocap **Property Data Type:** xsd:long **Multi-Valued:** NO

Description: The appID property indicates the application identifier of the application that owns the content. This value will match the application_id in the application_identifier signaling found in an AIT or XAIT. Specifically, this value encodes the organization and application Ids as (orgID << 16) | appID. This property is converted to an org.dvb.application.AppID object when accessed from the MetadataNode.getMetadata(String) method with a parameter of PROP_APP_ID.

Default Value: N/A – Output Only

Sort Order: Sequenced Numeric

Property Key: ocap:appID

C.1.1.1.11 Space Required

Namespace: ocap **Property Data Type:** xsd:int **Multi-Valued:** NO

Description: The spaceRequired property indicates the amount of space in bytes needed for the content.

Default Value: N/A – Output Only

Sort Order: Sequenced Numeric

Property Key: ocap:spaceRequired

C.1.1.1.12 URI for recorded content

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

Description: The contentURI property indicates the location of the content. This URI MAY be used for playback of recorded content. This URI corresponds to the <res> element created for the recording request. It is implementation-

specific regarding whether this property matches the ocap:destination property. An HNIMP SHALL format the value as defined for the A_ARG_TYPE_URI in the [UPNP CDS] specification.

Default Value: N/A – Output Only

Sort Order: Sequenced alpha-numeric

Property Key: ocap:contentURI

C.1.1.1.13 Media presentation point

Namespace: ocap **Property Data Type:** xsd:long **Multi-Valued:** NO

Description: The mediaPresentationPoint property indicates a resume media time in the content. The value is an offset in milliseconds from the beginning of the recording. An HNIMP does not set the value of this property except when a recording content item is created, at which point the HNIMP SHALL initialize the value to -1, which indicates not set. A control point MAY change the value; see Section 6.8.5.2.

Default Value: N/A – Output Only

Sort Order: Sequenced Numeric

Property Key: ocap:msoPresentationPoint

C.1.1.1.14 Indicator that content is MSO Recorded Content

Namespace: ocap **Property Data Type:** xsd:boolean **Multi-Valued:** NO

Description: The msoContentIndicator property indicates the content item is recorded content from a cable service. The value of true indicates the content is a recording, and the value of false indicates it is not content that was streamed, recorded, or copied from an MSO network.

Default Value: N/A – Output Only

Sort Order: Sequenced numeric

Property Key: ocap:msoContentIndicator

C.1.1.1.15 Expiration Period

Namespace: ocap **Property Data Type:** xsd:int **Multi-Valued:** NO

Description: The expiration property indicates the period in seconds the recording expires after being recording initiates.

Default Value: N/A – Output Only

Sort Order: Sequenced Numeric

Property Key: ocap:expiration

C.1.1.1.16 Media Storage Volume

Namespace: ocap **Property Data Type:** xsd:int **Multi-Valued:** NO

Description: The mediaStorageVolume property indicates the content item is a storage volume where recordings can be made.

Default Value: N/A – Output Only

Sort Order: N/A

Property Key: ocap:mediaStorageVolume

C.1.1.1.17 *First Media Time*

Namespace: ocap **Property Data Type:** xsd:long **Multi-Valued:** NO

Description: The first media time property is an offset in milliseconds to the first media time in the content. The HNIMP SHALL set this value to the same value returned by the RecordedService.getFirstMediaTime for the same content.

Default Value: N/A – Output Only

Sort Order: Sequenced alpha-numeric

Property Key: ocap:mediaFirstTime

C.1.1.1.18 *Channel Format*

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

Description: The ocap:scheduleChannelIDType property determines the format that is used for ocap:scheduledChannelID property as defined above. The definition of this property is same as srs:scheduledChannelID@type property. See [UPNP SRS] Appendix B.4.2.1.

NOTE (informative): The value of this property may be extended as "Vendor-defined" type in the future specification.

Default Value: N/A – Required on input.

Sort Order: Same as srs:scheduledChannelID@type.

Property Key: "ocap:scheduledChannelIDType"

C.1.2 Definitions of SRS properties

The following properties are used in the SRS objects.

C.1.2.1 *scheduledCDSEntryID*

Namespace: ocap **Property Data Type:** xsd:string **Multi-Valued:** No

Description: The scheduledCDSEntryID property contains the *didl-lite:@id* property value of the CDS object created for the NetRecordingEntry or the *didl-lite:@id* property value of the CDS object created for the RecordingContentItem with the associated recordTask.

Default Value: N/A – Required on input

Sort Order: Lexical or Lexical Numeric.

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

Input: The desired setting

Output: The current setting

PropertyKey: "ocap:scheduledCDSEntryID"

C.1.2.1.1 *cdsReference*

Namespace: ocap **Property Data Type:** xsd:string **Multi-Valued:** No

Description: The *cdsReference* property MUST only contain metadata of a CDS object that is referenced directly by a *recordSchedule* or *recordTask* object using *ocap:scheduledCDSEntryID* property.

The *cdsReference* property MUST contain a valid and properly escaped DIDL-Lite XML Document. Here is the example value of this property.

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL-Lite
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:upnp="urn:schemas-upnp-org:metadata-1-0/upnp/"
xmlns="urn:schemas-upnp-org:metadata-1-0/DIDL-Lite/">
<item id="RCI01" parentID="container001"
restricted="0">
<dc:title>Music Show</dc:title>
<upnp:class>
object.item.videoItem
</upnp:class>
<ocap:scheduledStartTime>20090503T19:00:00</ocap:scheduledStartTime>
<ocap:scheduledDuration>P01:00:00</ocap:scheduledDuration>
<ocap:scheduledChannelIDType>DIGITAL</ocap:scheduledChannelIDType>
<ocap:scheduledChannelID>5,1</ocap:scheduledChannelID>
<desc xmlns:ocapApp="urn:schemas-opencable-com:ocap-application
namespace="urn:schemas-opencable-com:ocap-application"
id="descriptorX">
<ocapApp:Band-Members>
<ocapApp:Guitar-Player> John Smith </ocapApp:Guitar-Player>
</ocapApp:Band-Members>
</desc>
</item>
</DIDL-Lite>
```

Note that this is NOT a multi-valued property, unlike *srs:cdsReference*. So, this property can be obviously associated with the *ocap:scheduledCDSEntryID* property that is included in the same *recordSchedule* or *recordTask*.

Default Value: N/A – Output only.

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

Input: N/A.

Output: The current setting

PropertyKey: "ocap:cdsReference"

C.1.3 Definitions of OCAP-extended properties of CDS item class for series recordings

The following properties are used to represent the relationship between a RecordingContentItem and a NetRecordingEntry. The ocap:netRecordingEntry property is included in the CDS entry corresponding to the RecordingContentItem. The ocap:RCIList property is included in the CDS entry corresponding to the NetRecordingEntry.

C.1.3.1.1 NetRecordingEntry ID

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

Description: The CDS object ID of the NetRecordingEntry to which the RecordingContentItem was added.

Default Value: N/A

Property Key: "ocap:netRecordingEntry"

C.1.3.1.2 RecordingContentItem list

Namespace: ocap **Property Data Type:** CVS (xsd:string) **Multi-Valued:** NO

Description: This property is composed of a comma-separated list of CDS object IDs of the RecordingContentItems that the NetRecordingEntry contains.

Default Value: N/A

Property Key: "ocap:RCIList"

C.2 Actions

UPnP actions are defined for OCAP requirements.

C.2.1 Recording X_PrioritizeRecordings

The X_PrioritizeRecordings action is a vendor action added by OpenCable to the UPnP Scheduled Recording Service, and an OC-DMS SHALL support this action. This action prioritizes a set of record schedules or a set of record tasks for resource usage. Recordings are ordered by RecordScheduleID or RecordTaskID. The order of appearance determines the priority with a recording appearing before another recording having priority. When a record schedule is listed without any record tasks, then all record tasks in that schedule will have priority over subsequently listed record schedules and tasks. This action causes a "best effort" on the server where non-existent RecordSchedule or RecordTask IDs are ignored, as is corresponding recording state.

C.2.1.1 **Arguments****Table C-3 - Arguments for X_PrioritizeRecordings()**

Argument	Direction	Related State Variable
RecordingIDs	IN	A_ARG_TYPE_ObjectIDList; see [UPNP CDS]

C.2.1.2 **Dependency on State**

None.

C.2.1.3 **Effect on State**

This action MAY cause changes in the state of RecordSchedule and RecordTasks objects.

C.2.1.4 **Errors****Table C-4 - Error Codes for X_Prioritize()**

errorCode	errorDescription	Description
400-499	Defined Elsewhere	See UPnP Device Architecture section on Control.
500-599	Defined Elsewhere	See UPnP Device Architecture section on Control.
600-699	Defined Elsewhere	See UPnP Device Architecture section on Control.
714	Non-homogenous IDs	X_PrioritizeRecordings() failed because the Object IDs specified by the RecordingIDs argument contains both RecordSchedule and RecordTask IDs.

Annex D Content TSPEC Population Requirements

The HNIMP SHALL populate the TSPEC associated with the content item in the CDS as specified in Annex C of [UPNP QMS]. The actual values used for various TSPEC parameters are specified in subsequent sections.

D.1 Default TSPEC values for Content Item

If no specific values are provided for TSPEC parameters of a content item, the HNIMP SHALL use the following default TSPEC parameter values depending on the type of a content item. See [UPNP QMS] for definition of TSPEC parameters mentioned below.

Table D-1 - Default TSPEC parameter values

TSPEC Parameter	High-Definition Content	Standard-Definition Content
PeakDataRate	20 Mbps	10 Mbps
DataRate	20 Mbps (same as Peak Data Rate, thus all streams are treated as Constant Bit Rate)	10 Mbps (same as PeakDataRate. Thus all streams are treated as Constant Bit Rate)
MaxBurstSize	0	0
MaxPacketSize	1500 bytes for HTTP Transport 1374 bytes for RTP Transport	1500 bytes for HTTP Transport 1374 bytes for RTP Transport
E2EMaxDelayHigh	500 ms	500 ms
E2EMaxJitter	165 ms	165 ms

Appendix I Complex Mapping Diagrams (informative)

A number of HN API method calls may invoke complex UPnP action mappings where more than one action occurs and where more than one device may be involved in a method mapping. The diagrams in this section demonstrate the interactions of such method calls.

I.1 Scheduling a Recording Sequence

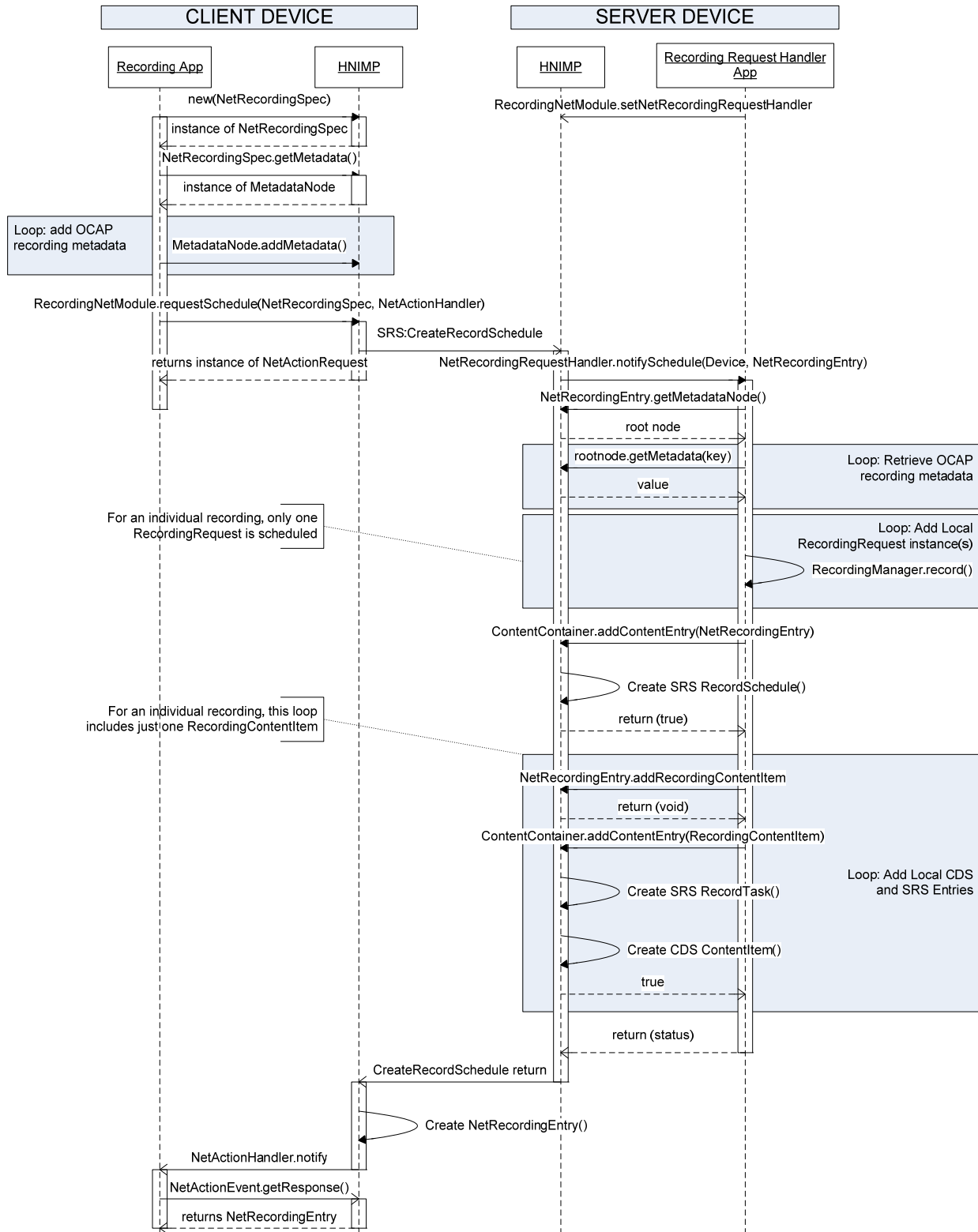


Figure I-1 - Recording Schedule Sequence Diagram 1

1. RecordingNetModule.setNetRecordingRequestHandler()

At some time, the Monitor Application or another privileged application registers a NetRecordingRequestHandler.

2. new(NetRecordingSpec)

The client Recording Application creates a new RecordingSpec instance.

3. NetRecordingSpec.getMetadata()

The client Recording Application retrieves the root metadata node for the NetRecordingSpec.

4. MetadataNode.addMetadata()

The client Recording Application loops as needed to add metadata sufficient to completely specify the desired recording, as required by the SRS CreateRecordSchedule method.

5. RecordingNetModule.requestSchedule(NetRecordingSpec, NetActionHandler)

The client Recording Application requests scheduling of recording from a RecordingNetModule that represents a DVR device, passing the new NetRecordingSpec instance and its NetActionHandler. This method returns an instance of a NetActionRequest which the client Recording Application can use to monitor the progress of the network request.

6. SRS:CreateRecordSchedule

The HNIMP constructs a record schedule request from the NetRecordingSpec provided by the client Recording Application, and passes this to the SRS in the server.

7. NetRecordingRequestHandler.notifySchedule(Device, NetRecordingEntry)

The HNIMP creates a NetRecordingEntry and populates the metadata node in this entry using the metadata received from the client in the CreateRecordSchedule method. Then the HNIMP calls the registered NetRecordingRequestHandler.notifySchedule method, passing the requested device instance and this constructed NetRecordingEntry.

8. NetRecordingEntry.getMetadataNode

The Recording Request Handler Application retrieves the metadata node provided by the HNIMP with the NetRecordingEntry.

9. getMetadata

The Recording Request Handler Application loops as needed to retrieve the metadata required to construct a RecordingRequest for the recording.

10. RecordingManager.record()

The Recording Request Handler Application creates the recording or series of recordings by interacting with the DVR API RecordingManager. The application is responsible for determining the components of a series recording, and may create the hierarchy of parent/leaf recordings.

11. ContentContainer.addContentEntry(NetRecordingEntry)

The Recording Request Handler Application calls ContentContainer.addContentEntry() with the NetRecordingEntry as a parameter, to have the HNIMP create a RecordSchedule.

12. NetRecordingEntry.addRecordingContentItem()

The Recording Request Handler Application calls NetRecordingEntry.addRecordingContentItem to associate the RecordingContentItem with the series recording represented by the NetRecordingEntry.

13. ContentContainer.addContentEntry(RecordingContentItem)

The Recording Request Handler Application adds the RecordingContentItem to the ContentContainer. The HNIMP then adds a RecordTask to the RecordSchedule associated with the NetRecordingEntry that contains the RecordingContentItem. The HNIMP also creates a content item in the CDS with cross-reference between the RecordTask and the content item.

14. SRS:CreateRecordSchedule return

The HNIMP sends the SRS:CreateRecordSchedule() method response back to the client. This includes the metadata about the scheduled recording.

15. new(NetRecordingEntry)

The HNIMP creates a NetRecordingEntry and populates it with the recording metadata, associating the response RecordScheduleID, Result and UpdateID with the NetRecordingEntry created on the client. This NetRecordingEntry is returned from NetActionEvent.getResponse.

16. NetActionHandler.notify(NetActionEvent)

The HNIMP calls the event notification method of the NetActionHandler passed by the client Recording Application to RecordingNetModule.requestSchedule(), passing an instance of NetActionEvent.

17. NetRecordingEvent.getResponse()

The client Recording Application retrieves the NetRecordingEntry that resulted from the request.

The client Recording Application can use the NetRecordingEntry to retrieve additional information about the scheduled recording. For example, to display the schedule status to the end user, Recording Application would perform these steps:

- a. Call ContentServerNetModule.requestSearchEntries() with SearchCriteria that includes a condition for the item that has RecordScheduleID associated with the NetRecordingEntry. The result is a list of RecordingContentItem.
- b. Call RecordingContentItem.getMetadata() to retrieve the root metadata node for each RecordingItem.
- c. Call MetadataNode.getMetadata() to retrieve PROP_RECORDING_STATE.

I.2 Cancel Individual Recording

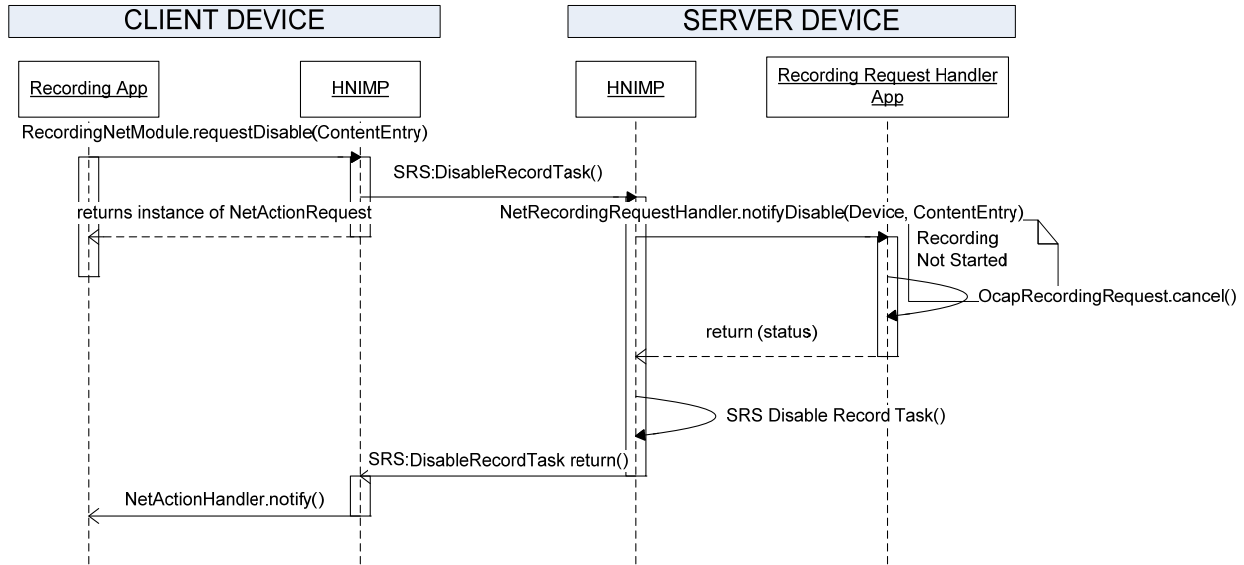


Figure I-2 - Cancel Individual Recording Sequence Diagram

1. RecordingNetModule.requestDisable(ContentEntry, NetActionHandler)

Recording Application requests that a scheduled recording be disabled. The network host is identified by the ContentEntry parameter. For RecordingNetModule.requestDisable(), if the recording is in progress, this method requests that the recording be stopped. If the recording is pending, this method requests that the recording be cancelled. This sequence diagram assumes that the recording has NOT started.

2. returns instance of NetActionRequest

The HNIMP returns an instance of a NetActionRequest which the Recording Application can use to monitor the progress of the network request, if desired.

3. SRS:DisableRecordTask()

The HNIMP constructs a disable record task request from the RecordingContentItem in the ContentEntry provided by Recording Application and passes this to the SRS in the recording server.

Instead of immediately modifying the SRS for the recording specified in the DisableRecordTask request, the server HNIMP uses the RecordTaskID received with the DisableRecordTask method to locate RecordingContentItem with the RecordTaskID for the recording. (If the request was attempting to cancel an entire series recording, or a one-time recording, the HNIMP could use the RecordScheduleID to locate the NetRecordingEntry.)

4. NetRecordingRequestHandler.notifyDisable(Device, ContentEntry)

The implementation calls the registered NetRecordingRequestHandler, passing the requested device instance and this retrieved RecordingContentItem, referenced through the ContentEntry interface.

The Recording Request Handler Application retrieves the metadata node provided by the HNIMP with this `NetRecordingEntry`, and can use the `RecordingManager.getEntries()`, filtering based on the recording metadata, to retrieve the `OcapRecordingRequest`. It can check the status of the recording request.

5. `OcapRecordingRequest.cancel()`

Because the recording is NOT in progress, Recording Request Handler Application cancels the recording request.

6. `SRS:DisableRecordTask` return

The HNIMP provides status back to the client.

7. `NetActionHandler.notify(NetActionEvent)`

The HNIMP calls the event notification method of the `NetActionHandler` passed by the client Recording Application to `RecordingNetModule.requestSchedule()`, passing an instance of `NetActionEvent`. In this case, `NetActionEvent.getResponse()` returns null. The result of processing can be obtained by `NetActionEvent.getActionStatus()`. If processing failed, `NetActionEvent.getError()` returns the error number.

I.3 Stop Individual Recording

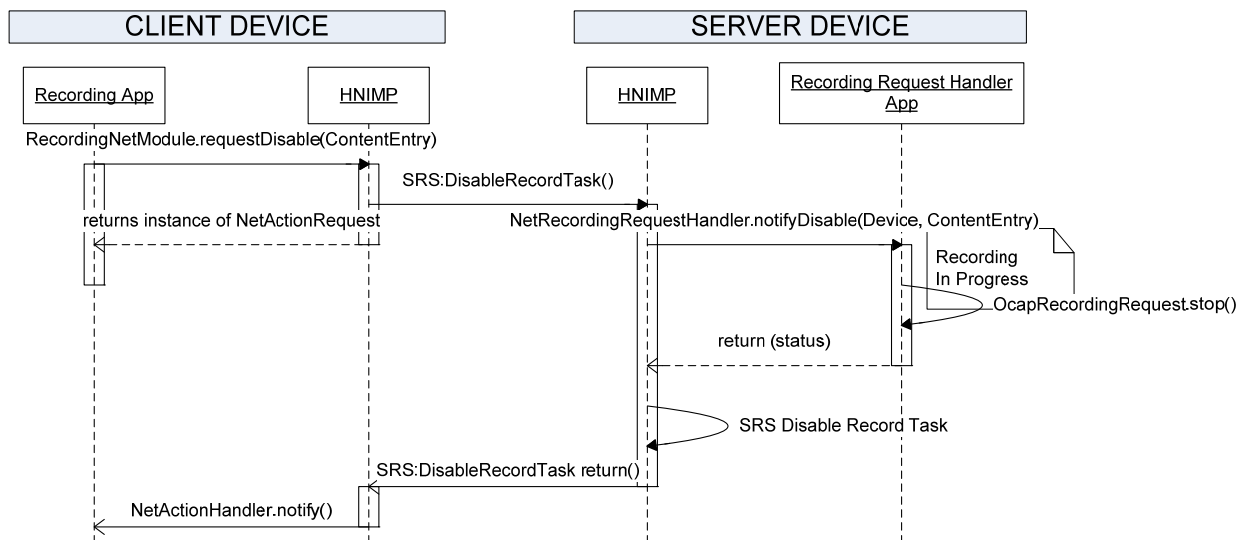


Figure I-3 - Stop Individual Recording Diagram

1. `RecordingNetModule.requestDisable(ContentEntry, NetActionHandler)`

Recording Application requests that a scheduled recording be disabled. The network host is identified by the `ContentEntry` parameter. For `RecordingNetModule.requestDisable()`, if the recording is in progress, this method requests that the recording be stopped. If the recording is pending, this method requests that the recording be cancelled. This sequence diagram assumes that the recording has started.

2. returns instance of NetActionRequest

The HNIMP returns an instance of a NetActionRequest which the Recording Application can use to monitor the progress of the network request, if desired.

3. SRS:DisableRecordTask()

The HNIMP constructs a disable record task request from the RecordingContentItem in the ContentEntry provided by Recording Application and passes this to the SRS in the recording server.

Instead of immediately modifying the SRS for the recording specified in the DisableRecordTask request, the server HNIMP uses the RecordTaskID received with the DisableRecordTask method to locate RecordingContentItem with the RecordTaskID for the recording.

4. NetRecordingRequestHandler.notifyDisable(Device, ContentEntry)

The implementation calls the registered NetRecordingRequestHandler, passing the requested device instance and this RecordingContentItem.

The Recording Request Handler Application retrieves the metadata node provided by the HNIMP with this NetRecordingEntry, and can use the RecordingManager.getEntries(), filtering based on the recording metadata, to retrieve the OcapRecordingRequest. It can check the status of the recording request.

5. OcapRecordingRequest.stop()

Because the recording is in progress, Recording Request Handler Application stops the recording request.

6. SRS:DisableRecordTask return

The HNIMP provides status back to the client.

7. NetActionHandler.notify(NetActionEvent)

The HNIMP calls the event notification method of the NetActionHandler passed by the client Recording Application to RecordingNetModule.requestSchedule(), passing an instance of NetActionEvent. The result of processing can be obtained by NetActionEvent.getActionStatus(). If processing failed, NetActionEvent.getError() returns the error number.

I.4 Delete Individual Recording

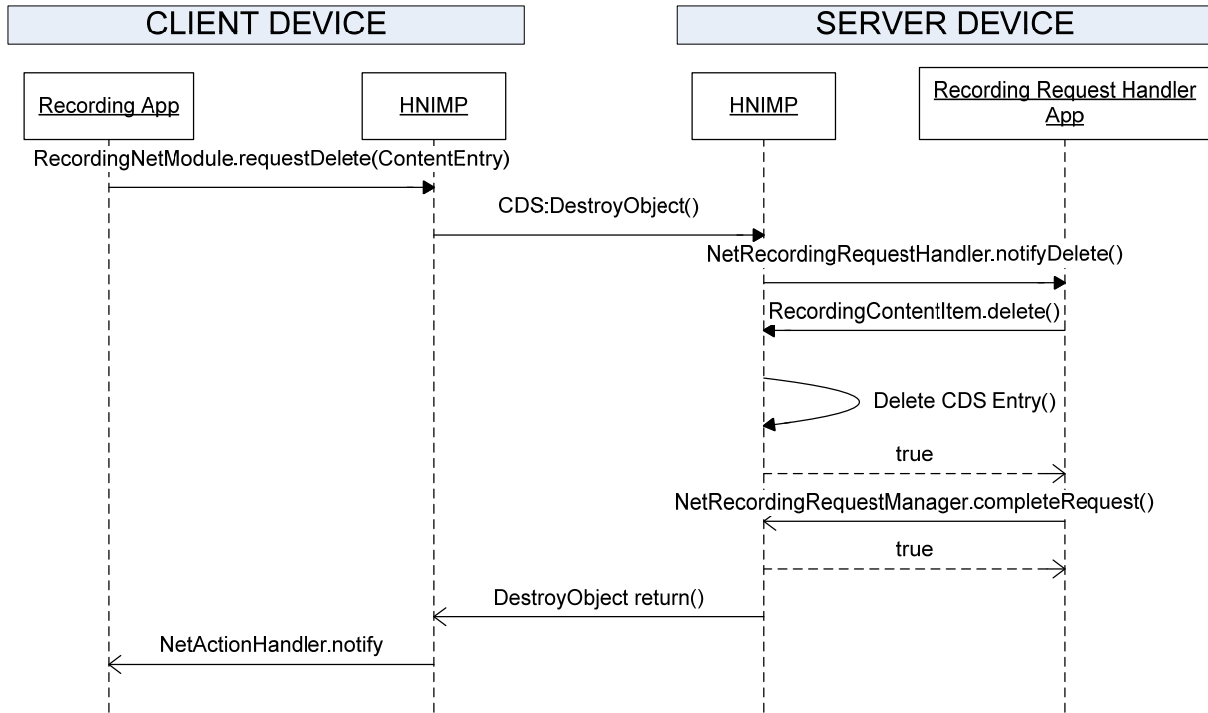


Figure I-4 - Delete Individual Recording Diagram

I.5 Schedule Local Recording

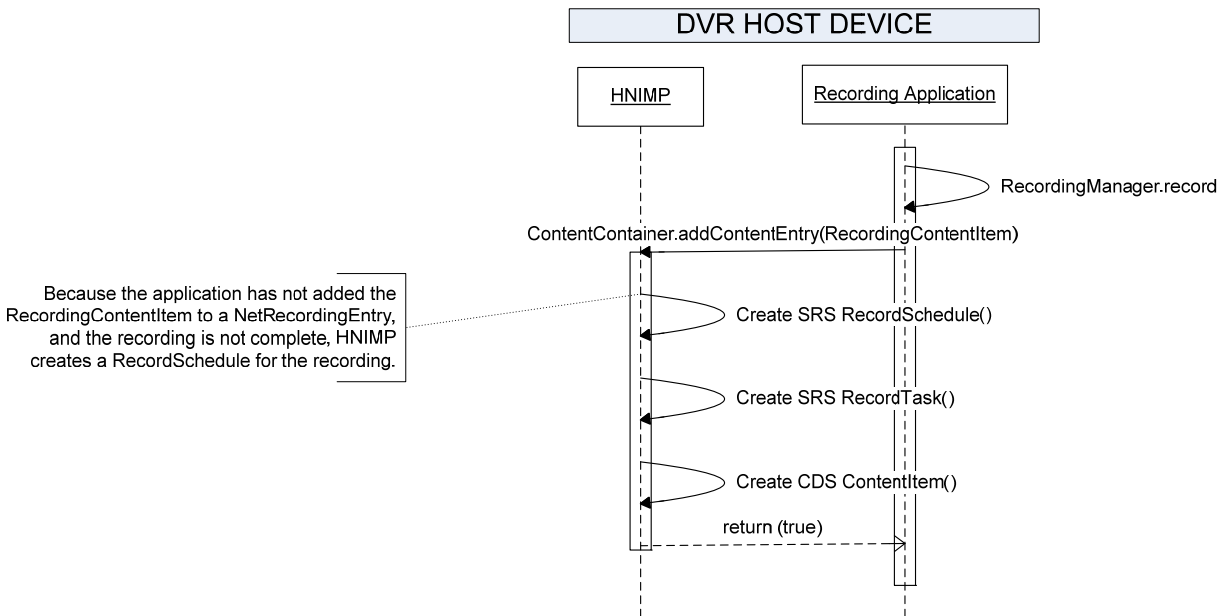


Figure I-5 - Schedule Local Recording Diagram

1. RecordingManager.record()

The Recording Application in a DVR host device schedules a recording.

2. ContentContainer.addContentEntry(RecordingContentItem)

Although the Recording Request Handler Application notifySchedule() method is required to create a NetRecordingEntry for remote recording requests, the local Recording Application is not required to create a NetRecordingEntry. It can expose the recording by adding the RecordingContentItem to ContentContainer.

Because the RecordingContentItem is not contained in a NetRecordingEntry and has no associated RecordSchedule, the HNIMP creates the RecordSchedule, creates the RecordTask, and also creates the CDS ContentItem.

I.6 Add Individual Recording to Series Recording

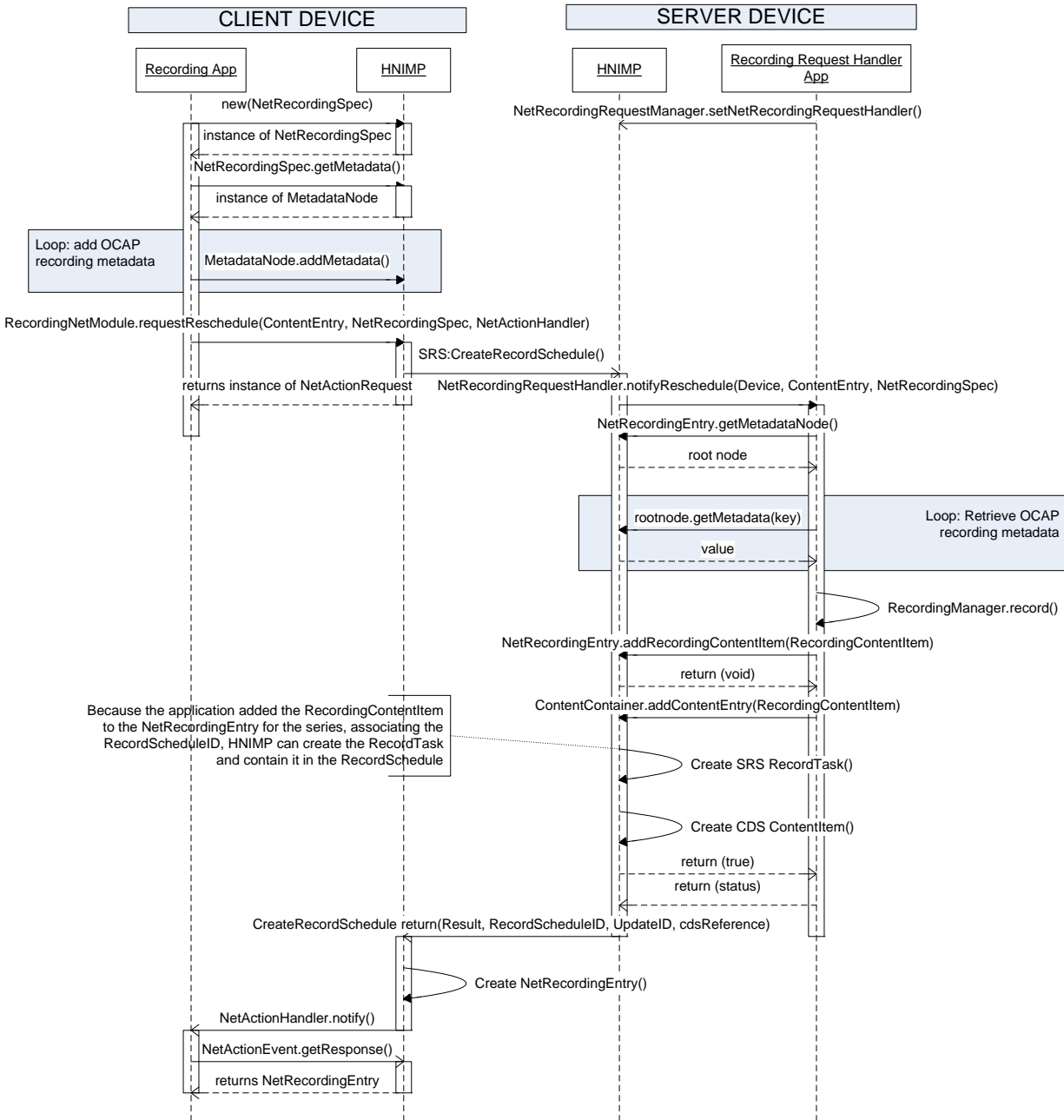


Figure I-6 - Add Recording to Series Diagram

1. RecordingNetModule.setNetRecordingRequestHandler()

At some time, the Monitor Application or another privileged application registers a NetRecordingRequestHandler.

2. new(NetRecordingSpec)

The client Recording Application creates a new RecordingSpec instance.

3. NetRecordingSpec.getMetadata()

The client Recording Application retrieves the root metadata node for the NetRecordingSpec.

4. MetadataNode.addMetadata()

The client Recording Application loops as needed to add metadata sufficient to completely specify the desired recording, as required by the SRS CreateRecordSchedule method.

5. RecordingNetModule.requestReschedule(NetRecordingSpec, NetActionHandler)

The client Recording Application requests changing the scheduled recording from a RecordingNetModule that represents a DVR device, passing the existing ContentEntry representing the series recording, the new NetRecordingSpec instance for the recording to be added to the series, and its NetActionHandler. This method returns an instance of a NetActionRequest which the client Recording Application can use to monitor the progress of the network request.

6. SRS:CreateRecordSchedule

The HNIMP constructs a record schedule request from the NetRecordingSpec provided by the client Recording Application, and passes this to the SRS in the server.

7. NetRecordingRequestHandler.notifyReschedule(Device, NetRecordingEntry)

The HNIMP uses the metadata from the request received from the client in the CreateRecordSchedule method to locate the existing ContentEntry that represents the series recording. It also constructs a NetRecordingSpec representing the recording to be added to the series. Then the HNIMP calls the registered NetRecordingRequestHandler.notifySchedule method, passing the requested device instance, the retrieved series ContentEntry, and the new NetRecordingSpec.

8. NetRecordingEntry.getMetadataNode

The Recording Request Handler Application retrieves the metadata node provided by the HNIMP with the NetRecordingEntry.

9. getMetadata

The Recording Request Handler Application loops as needed to retrieve the metadata required to construct a RecordingRequest for the recording.

10. RecordingManager.record()

The Recording Request Handler Application creates the recording by interacting with the DVR API RecordingManager.

11. NetRecordingEntry.addRecordingContentItem()

The Recording Request Handler Application identifies the NetRecordingEntry that represents the series recording for the new recording, either by examining metadata, finding the ParentRecordingRequest of the new

RecordingRequest (not shown). Then it calls `NetRecordingEntry.addRecordingContentItem` to associate the `RecordingContentItem` with the series recording represented by this `NetRecordingEntry`.

12. `ContentContainer.addContentEntry(RecordingContentItem)`

The Recording Request Handler Application adds the `RecordingContentItem` to the `ContentContainer`. The HNIMP then adds a `RecordTask` to the `RecordSchedule` associated with the `NetRecordingEntry` that contains the `RecordingContentItem`. The HNIMP also creates a content item in the CDS with cross-reference between the `RecordTask` and the content item.

13. `SRS:CreateRecordSchedule` return

The HNIMP sends the `SRS:CreateRecordSchedule()` method response back to the client. This includes the metadata about the scheduled recording.

14. `new(NetRecordingEntry)`

The HNIMP creates a `NetRecordingEntry` and populates it with the recording metadata, associating the response `RecordScheduleID`, `Result` and `UpdateID` with the `NetRecordingEntry` created on the client. This `NetRecordingEntry` is returned from `NetActionEvent.getResponse`.

15. `NetActionHandler.notify(NetActionEvent)`

The HNIMP calls the event notification method of the `NetActionHandler` passed by the client Recording Application to `RecordingNetModule.requestSchedule()`, passing an instance of `NetActionEvent`.

16. `NetRecordingEvent.getResponse()`

The client Recording Application retrieves the `NetRecordingEntry` that resulted from the request.

The client Recording Application can use the `NetRecordingEntry` to retrieve additional information about the scheduled recording. For example, to display the schedule status to the end user, Recording Application would perform these steps:

- a. Call `ContentServerNetModule.requestSearchEntries()` with `SearchCriteria` that includes a condition for the item that has `RecordScheduleID` associated with the `NetRecordingEntry`. The result is a list of `RecordingContentItem`.
- b. Call `RecordingContentItem.getMetadata()` to retrieve the root metadata node for each `RecordingItem`.
- c. Call `MetadataNode.getMetadata()` to retrieve `PROP_RECORDING_STATE`.

Appendix II Revision History

The following ECNs were incorporated into version I02 of this specification:

EC Identifier	Accepted Date	Title of EC
HNP2.0-N-09.1482-1	12/17/09	Clarifications about client and server trick mode behavior
HNP2.0-N-09.1472-1	12/17/09	Remove ParentRecordingRequest state mapping
HNP2.0-N-09.1467-1	12/17/09	Clarify that recording entry objectID is a CDS ObjectID
HNP2.0-N-09.1466-1	12/17/09	NetActionEvent.getError mapping clarification
HNP2.0-N-09.1461-1	12/17/09	AudioResource language mapping fix
HNP2.0-N-09.1453-2	12/17/09	Fix inconsistencies in ObjectID handling
HNP2.0-N-09.1445-2	12/17/09	Corrections and clarifications in the OC-DMS device description regarding supported services
HNP2.0-N-09.1439-2	12/17/09	Event subscription clarification
HNP2.0-N-09.1433-1	12/17/09	Clarification about use of ocap namespace for OCAP properties defined in Annex C
HNP2.0-N-09.1432-1	12/17/09	Corrections to make consistent use of "scheduledStartDateTime" and "ocapApp"
HNP2.0-N-09.1431-1	12/17/09	Mandate server support for Range headers
HNP2.0-N-09.1426-1	12/17/09	Update OCAP Reference
HNP2.0-N-09.1393-2	12/17/09	Map new method Device.setProperties
HNP2.0-N-09.1369-3	12/17/09	Set RecordingContentItem class to videoItem
HNP2.0-N-08.1343-10	12/17/09	UPnP Object Clarifications
HNP2.0-N-08.1342-1	12/17/09	Remove ContentDatabase class from HNP
HNP2.0-N-08.1341-1	12/17/09	Remove ContentManagementNetModule interface from HNP2.0 and clarify support CDS actions
HNP2.0-N-08.1340-3	12/17/09	Clarify SRS related methods in HNP2.0
HNP2.0-N-08.1334-5	12/17/09	Clarify processing of Schedule Recording
HNP2.0-N-08.1314-5	12/17/09	Revise informative appendix sequence diagrams and narrative
HNP2.0-N-08.1307-1	10/30/08	HN MSO Content Indicator property fix
HNP2.0-N-08.1299-7	12/17/09	HN Authorization API Mapping
HNP2.0-N-08.1298-3	12/17/09	Application Metadata in RecordingContentItem
HNP2.0-N-08.1289-1	12/17/09	Mandatory Media Format Profile fixes
HNP2.0-N-08.1257-2	7/14/08	Home Networking Recording API mapping fixes
HNP2.0-N-08.1254-1	7/14/08	Device IP Address Mapping

The following ECNs were incorporated into version I03 of this specification:

EC Identifier	Accepted Date	Title of EC
HNP2.0-N-09.1484-1	6/3/10	ContentEntry.getCreationDate() clarification
HNP2.0-N-10.1544-1	6/3/10	HNP2.0 Home Networking Asset In Use Detection
HNP2.0-N-10.1548-1	6/3/10	Synchronizing PresentationPoint and MediaTime for persistent bookmarking

