OpenCable™ Specifications
Home Networking 2.0

Home Networking Protocol 2.0

OC-SP-HNP2.0-I10-130530

ISSUED

Notice

DISCLAIMER

This document is published by Cable Television Laboratories, Inc. ("CableLabs®").

CableLabs reserves the right to revise this document for any reason including, but not limited to, changes in laws, regulations, or standards promulgated by various agencies; technological advances; or changes in equipment design, manufacturing techniques, or operating procedures described, or referred to, herein. CableLabs makes no representation or warranty, express or implied, with respect to the completeness, accuracy, or utility of the document or any information or opinion contained in the report. Any use or reliance on the information or opinion is at the risk of the user, and CableLabs shall not be liable for any damage or injury incurred by any person arising out of the completeness, accuracy, or utility of any information or opinion contained in the document.

This document is not to be construed to suggest that any affiliated company modify or change any of its products or procedures, nor does this document represent a commitment by CableLabs or any cable member to purchase any product whether or not it meets the described characteristics. Nothing contained herein shall be construed to confer any license or right to any intellectual property, whether or not the use of any information herein necessarily utilizes such intellectual property. This document is not to be construed as an endorsement of any product or company or as the adoption or promulgation of any guidelines, standards, or recommendations.

Document Status Sheet

Document Control Number:	OC-SP-HNP2.0-I10-130530			
Document Title:	Home Networking Protocol 2.0			
Revision History:	I01 - Released 4/18/08 I02 - Released 12/17/09 I03 - Released 6/3/10 I04 - Released 2/24/11 I05 - Released 5/12/11 I06 - Released 1/12/12 I07 - Released 2/24/12 I08 - Released 5/31/12 I09 - Released 4/18/13 I10 - Released 5/30/13			
Date:	May 30, 2013			
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

Contents

Figures

Tables

This page left blank intentionally.

1 SCOPE

1.1 Introduction and Overview

The OpenCable Home Networking Protocol defines the minimum requirements based on the Host Home Networking 2.0 Extension [HOSTHN] and the OCAP Home Networking Extension [OCAP-HN]. This specification 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.

All references are subject to revision, and parties to agreement based on this specification are encouraged to investigate the possibility of applying the most recent editions of the documents listed below.

References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific:

- For a specific reference, subsequent revisions do not apply.
- For a non-specific, non-Bundle reference, the latest version applies.
- For non-specific CableLabs references that are part of the [OC-BUNDLE], the versions mandated in a particular Bundle apply.

[CEA-766] CEA-766-C (ANSI): U.S. and Canadian Region Rating Tables (RRT) and Content Advisory Descriptors for Transport of Content Advisory Information Using ATSC Program and System Information Protocol (PSIP), April 2008.

[DLNA CR-214] DLNA Change Request 00214: Clarification of MPEG-2 bitstream compliance during server side trick mode playback (Server Side Trick Modes), May 16, 2012.

[DLNA Diag] Digital Living Network Alliance Diagnostics Guidelines v0.70.

[DLNAe2] DLNA 1.5 Guidelines Errata Volume 2, August 14, 2007.

[DLNA RUIH] Digital Living Network Alliance HTML5 Guidelines v0.85 https://members.dlna.org/apps/org/workgroup/tc-html5-tf/download.php/25176/HTML-5_RUI_Guidelines_0_85.doc

[DLNA vol 1] Digital Living Network Alliance Home Networked Device Interoperability Guidelines; expanded: October 2006, Volume 1: Architectures and Protocols.

[DLNA vol 2] Digital Living Network Alliance Home Networked Device Interoperability Guidelines; expanded: December 2011, Volume 2: Media Format Profiles.

[DLNA vol 3] Digital Living Network Alliance Home Networked Device Interoperability Guidelines, expanded: October 2006, Volume 3: Link Protection.

[DLNA vol 5] Digital Living Network Alliance Home Networked Device Interoperability Guidelines, Volume V: Device Profiles.

[DLNA 3D MF] Digital Living Network Alliance 3D Media Formats: referenced to [DLNA vol 2].

[DLNA HTTP-AD] Digital Living Network Alliance HTTP Adaptive Delivery Guidelines v.0.50.

[DTCP-IP] DTCP Volume 1 Supplement E Mapping DTCP to IP (Informational Version), version 1.2, Jun 15 2007.

[HN-SEC] Home Networking Security Specification, OC-SP-HN-SEC, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].

[HOST] OpenCable Host Device 2.1 Core Functional Requirements, OC-SP-HOST2.1, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].

[MIB-HN] OpenCable Home Networking MIB Specification, OC-SP-MIB-HN, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].

[MPEG2-1] ISO/IEC 13818-1:2000 Information technology -- Generic coding of moving pictures and associated audio information: Systems, International Standards Organization.

[OC-BUNDLE] OC-SP-BUNDLE, OpenCable Bundle Requirements. See Section 2.3.1 to acquire this specification.

[OCAP-DVR] OCAP Digital Video Recorder (DVR), OC-SP-OCAP-DVR, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].

[OCAP-HN] OpenCable Host Home Networking Extension 2.0, OC-SP-HOST-HN2.0, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].

[OCAP] OpenCable Application Platform (OCAP), OC-SP-OCAP, Cable Television Laboratories, Inc. Referenced in [OC-BUNDLE].

[RFC 2045] IETF RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, N. Freed, N. Borenstein, November 1996.

[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 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 RUISS] UPnP RemoteUIServer v1, RemoteUIServer:1 Service Template Version 1.01, UPnP Forum, September 2, 2004.

[UPNP SRS] UPnP ScheduledRecording v1, ScheduledRecording:1 Service Template Version 1.01, UPnP Forum, September 19, 2007.

[XML] XML W3C Recommendations: Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, October 6, 2000.

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.

[DVB-MHP 1.1.3] ETSI TS 102 812 V1.3.1, DVB Multimedia Home Platform 1.1.3, DVB-MHP 1.1.3, Blue book A068r3.

[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 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 NS] Namespaces in XML 1.0 (Second Edition), W3C Recommendation, 16 August 2006.

2.3 Reference Acquisition

2.3.1 OpenCable Bundle Requirements

The OpenCable Bundle Requirements specification [OC-BUNDLE] indicates the set of CableLabs specifications required for the implementation of the OpenCable Bundle. The version number of [OC-BUNDLE] corresponds to the release number of the OpenCable Bundle that it describes. One or more versions of [OC-BUNDLE] reference this specification. Current and past versions of [OC-BUNDLE] may be obtained from CableLabs at http://www.cablelabs.com/opencable/specifications.

2.3.2 Other References

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027;
  Phone +1-303-661-9100; Fax +1-303-661-9199; Internet: http://www.cablelabs.com/
- Digital Living Network Alliance (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.
Broadcast Content	Any MSO Content streamed from an HFC network, and received by using the tuner of an OCHN Device.
Recorded Content	Broadcast content that is stored to disk as a permanent recording (e.g., an instance of OcapRecordingRequest).
Time-shifted content	Broadcast content that is stored temporarily to disk for the purposes of time-shifted presentation.
Live Content	Broadcast content that is not time shifted.
OpenCable Bundle	The OpenCable Bundle defines a set of specifications required to build a specific version of an OpenCable device. See [OC-BUNDLE].

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.

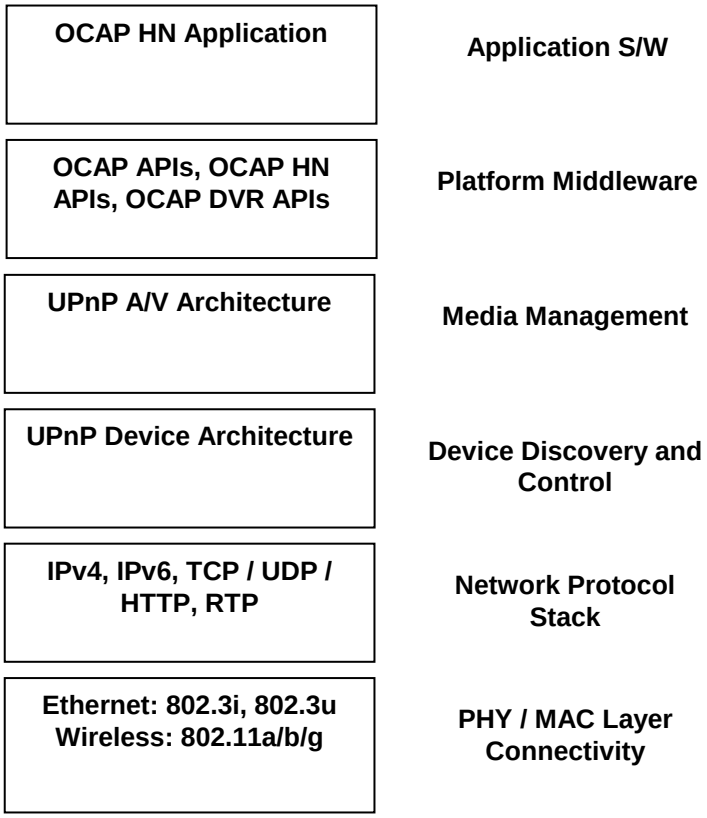

Figure 5–1 - OCAP Home Networking Framework for UPnP

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

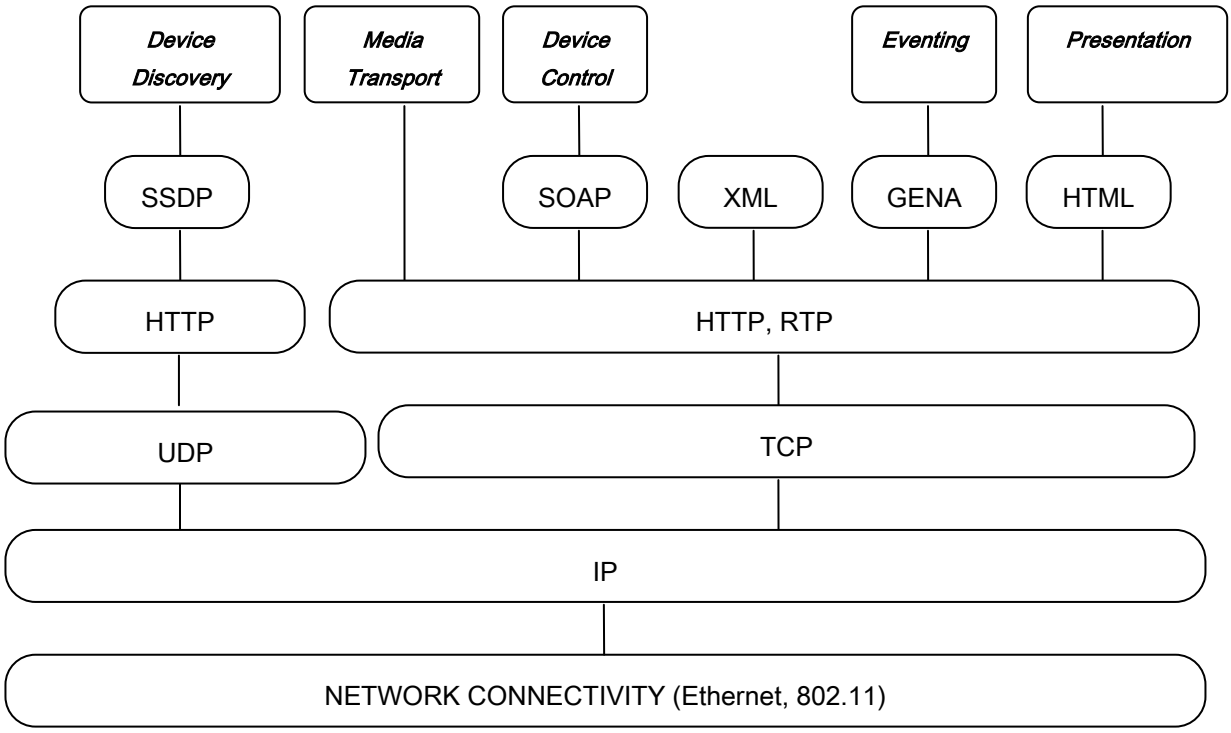

Figure 5–2 - UPnP Device Architecture Protocol Stack

5.2 DLNA Compliance

The HNIMP definition is based on the Digital Living Network 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] 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]; and
- operate under a Uniform Client Data Availability Model (UCDAM) (see section 7.4 of [DLNA vol 1]);
- support "Full Random Access Data Availability" model (see guideline 7.4.15 of [DLNA vol 1]) for recorded content, including in-progress recordings;

- support "Limited Random Access Data Availability" Model Mode 0 (see guideline 7.4.16 of [DLNA vol 1]) for trick play of channel content items. (The HNIMP OC-DMS need not support trick play if [OCAP-DVR] is not supported.)

NOTE: The HNIMP will be operating under increasing S_N model for in progress recordings, and under the increasing S_0 model for time-shifted content.

When an HNIMP supports the DMS device class, the HNIMP SHALL support the +RUIHSRC+ capability defined in [DLNA RUIH] with the following requirements:

- The HNIMP SHALL support the UIListingUpdate evented state variable in [UPNP RUISS].
- If the HNIMP serves an RUI-H application, the RUI-H Transport Server (RUIHTS) device function SHALL be mandatory; otherwise, the RUI-H Transport Server is optional.

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.

5.3.1 Persistent Link-Local Addressing

Typically per [UPNP DA] an Auto-IP address is only acquired when no routable IP address can be acquired by DHCP and released if/when DHCP-based address acquisition is successful. These transitions between link-local and DHCP-assigned routable addresses can result in dropped streaming sessions and other anomalies. Persistent Link-Local Addressing is a condition where the HNIMP maintains a link-local address (i.e., a non-routable address in the range 169.254.x.x obtained via Auto-IP) regardless of whether an interface has acquired a routable IP address via DHCP [RFC 3927]. Persistent Link-Local Addressing is indicated by the ocHnNetConfigPersistentLinkLocalAddress MIB defined in [MIB-HN]. The HNIMP SHALL query the value of this MIB object at the beginning of normal operation (see [OCAP] "Boot Process"). If the value of the ocHnNetConfigPersistentLinkLocalAddress MIB object is true, the HNIMP SHALL acquire a link-local IP address as defined by [UPNP DA] sections 0.2 and 0.3, even if an IP address has been received using DHCP. If the value of the ocHnNetConfigPersistentLinkLocalAddress MIB object is true and a DHCP-assigned address is received after a link-local address is acquired, the HNIMP SHALL retain the link-local address. If the value of the ocHnNetConfigPersistentLinkLocalAddress MIB object is false or undefined, the HNIMP SHALL follow [UPNP DA] section 0.1 to determine when a link-local address needs to be acquired. The HNIMP SHALL ignore any changes to the ocHnNetConfigPersistentLinkLocalAddress MIB object after the start of normal operation.

5.3.1.1 HNIMP OC-DMS Requirements

When an HNIMP OC-DMS is configured with multiple IP addresses, it SHALL send UPnP advertisements for these IP addresses with a discreet unique device number (UDN) for each address. See [UPNP DA] for UDN definition and usage. M-SEARCH request responses SHALL incorporate the UDN corresponding to the subnet of the source IP address of the request. If the request does not correspond to any subnet configured on the DMS, then the OC-DMS SHALL issue an M-SEARCH response incorporating the UDN associated with the link-local address if the value of the ocHnNetConfigPersistentLinkLocalAddress MIB object is true. If the value is false or undefined, the OC-DMS SHALL issue the response incorporating a UDN associated with any OC-DMS configured address. To improve interoperability with control points that have difficulty processing UPnP messages containing link-local addresses in a mixed-IP network, UPnP advertisement messages (ssdp:alive, update, and byebye) and M-SEARCH response messages for link-local addresses SHALL be issued after the corresponding non-link-local message when both address types are utilized.

If the HNIMP OC-DMS receives a CDS:Browse or CDS:Search SOAP request with the ALLIP filter value omitted, the HNIMP OC-DMS SHALL incorporate res URI values corresponding to the subnet of the destination IP address of the request into the request response. See [DLNA vol 1] Annex C for more information.

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.

When an HNIMP supports the DMS and/or DMP device class, it SHALL comply with [DLNA vol 1] guideline 7.1.29.1. In guideline 7.1.29.1, the word “should” is replaced with the normative “SHALL”.

When an HNIMP supports simultaneous operation of MoCA and Ethernet, or MoCA and 802.11 interfaces, then the guidelines in [DLNA vol 1] annexes A.2.23, A.2.24, and A.2.25 are considered mandatory and the HNIMP SHALL comply with them. Within annexes A.2.23, A.2.24, and A.2.25, all occurrences of the word “should” are replaced with the normative “SHALL”

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
- GetFeatureList

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

An HNIMP OC-DMS SHALL use the same port number in the <res> value (URI) for all audio and video content items added to a ContentDirectory service that are streamed over HTTP.

NOTE (Informative): The HNIMP OC-DMS should allow the port number to be specified using some boot time or deployment time configuration and should attempt to use that port number first. If that port number cannot be used for any reason (e.g., the port is in use), then the OC-DMS should select an alternate port number.

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, ocap:scheduledChannelID, or srs:scheduledChannelID is used and the type is SI, the value SHALL be as specified in section B.4.2 of [UPNP SRS] and the fields SHALL be populated as follows:

- Network ID: any value (including the empty string) may appear; the following comma is required. This field is not used by this specification.
- Transport Stream ID: any value (including the empty string) may appear; the following comma is required. This field is not used by this specification.
- Service ID: the source_id signaled in a Virtual Channel Table.

Examples:

1. <upnp:channelID>,,3</upnp:channelID>
2. <upnp:channelID>0,0,42</upnp:channelID>

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

The HNIMP SHALL consider the namespaces specified in Table 1-3 of [UPNP CDS] as standardized namespaces for this specification.

In addition, 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 [OC-BUNDLE] version, e.g., 1.2.0. The value of this element matches the [OC-BUNDLE] version supported, e.g., "1.2.0". An example of <ocap:X_MiddlewareProfile> element is shown as follows:

```
<ocap:X_MiddlewareProfile xmlns:ocap="urn:schemas-cablelabs-com:device-1-
0">1.2.0</ 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>
```

5.5.4 CDS Features

If any ChannelGroupContainer has been published in the CDS, then the HNIMP SHALL respond to the CDS:GetFeatureList action with indication of support of the TUNER feature as defined in appendix E.2 of [UPNP CDS].

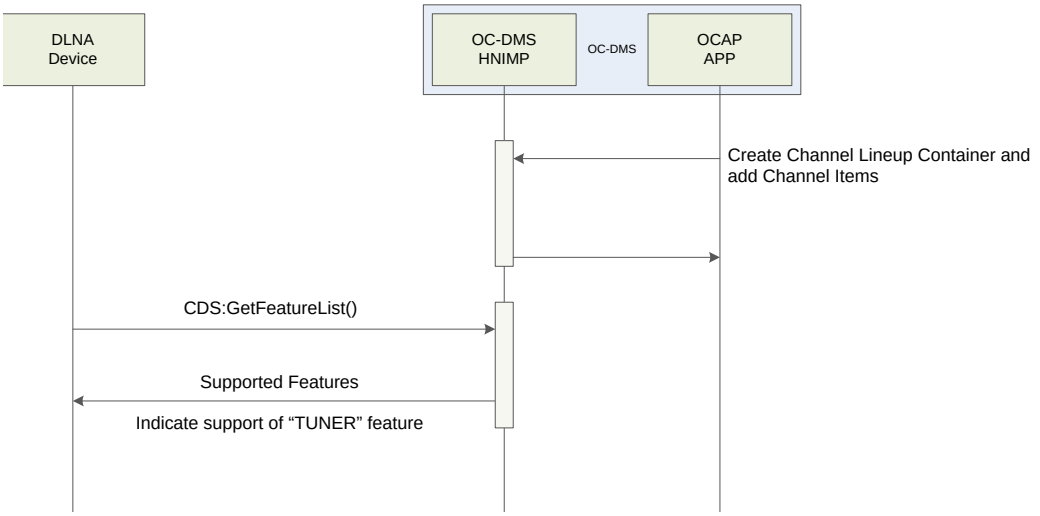

Figure 5–3 - Tuner Feature

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 on all recorded or Time-Shifted Content; 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 on all recorded or Time-Shifted Content; see [DLNA vol 1] section 7.4.70.

When an HNIMP is a DMS device, it SHALL support the Range HTTP header field on all recorded or Time-Shifted 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 on all recorded or Time-Shifted 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 the ServerSidePacedStreaming header is provided by the client to perform trick modes, then the HNIMP OC-DMS SHOULD attempt to re-stamp the PTS and PCR in the MPEG2-TS stream.

For trick mode support, an HNIMP OC-DMS that does not re-stamp the PTS/PCR in the stream SHALL,

- Use the FrameRateInTrickMode.ochn.org header if the client has sent the FrameRateInTrickMode.ochn.org header in the trick mode request.
- Use the FrameRateInTrickMode.dlna.org header as defined in [DLNA CR-214] if the client has not sent the FrameRateInTrickMode.ochn.org header in the trick mode request.

The HNIMP OC-DMS MAY keep the discontinuities in the resulting re-stamped content if there is a time stamp discontinuity in the original stream or when jumping across segments of a segmented recording. In the case of a segmented recording, the HNIMP OC-DMS MAY restart the re-stamping with the PTS and PCR values of the segment when transitioning from one segment to another.

Note (Informative): The HNIMP OC-DMS is expected to use the FrameRateInTrickMode.dlna.org header to communicate the rate at which the client is expected to play the frames in order to provide the requested trick mode operation as per [DLNA CR-214]. However, in order to provide backward compatibility, the HNIMP OC-DMS is expected to use the FrameRateInTrickMode.ochn.org header for clients that have explicitly used it in the request. In cases where the client has sent the ServerSidePacedStreaming header in the request and the server is not re-stamping the PTS/PCR, the server should use the FrameRateInTrickMode.dlna.org header.

During trick mode operations, the HNIMP client device SHALL be able to playback streams with decrementing PTS and 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
CurrentDecodePTS.ochn.org	Client→Server	optional	Trick
ChunkEncodingMode.ochn.org	Client→Server	required	Trick
MaxGOPsPerChunk.ochn.org	Client↔Server	required	Trick
MaxFramesPerGOP.ochn.org	Client→Server	optional	Trick
ServersidePacedStreaming.ochn.org	Client↔Server	optional	Trick
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 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

5.6.1.2 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.

5.6.1.3 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.

5.6.1.4 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 effect 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

5.6.1.5 ServersidePacedStreaming

An HNIMP OC-DMP MAY use the ServersidePacedStreaming header to request a server to enable pacing of MPEG data based on time stamps in the content. If the HNIMP client sends a trick play request including the ServersidePacedStreaming header, then the server SHOULD respond with an MPEG stream where both the PCR time stamp in the TS header's adaptation field and the PTS time stamp in the PES header are re-stamped, such that a client device that displays each frame as soon as its time stamp matures will play out at the requested trick play rate.

An HNIMP OC-DMS MAY support the ServersidePacedStreaming.ochn.org header. 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

If the server supports ServersidePacedStreaming and performs restamping for this request, then it SHALL return the ServersidePacedStreaming header in the response. If it does NOT perform restamping for this request, then it SHALL NOT return the ServersidePacedStreaming header in the response.

Note (Informative):

As an example, let us assume for simplicity that the original stream is 30fps and that the GOP size is 30 (fixed GOPs, with 1 I-frame per second). If the client asks for 10x FF playback, and the server responds with a stream consisting of I-frames that are 5 seconds apart in the original stream, and the first frame returned happens to have the time stamp 300, then the original (unmodified) sequence of time stamps for that I-frame data would be

300, 305, 310, 315, 320, 325, 330, 335, 340, 345 . . .

(These times are represented in seconds for simplicity of presentation; the actual time stamps would be in units of 27 MHz or 90 kHz.)

The requested MODIFIED time stamps for this sequence of frames would be:

300, 300.5, 301, 301.5, 302, 302.5, 303, 303.5, 304, 304.5 . . .

Thus, playing these frames according to their time stamps would result in each frame being displayed for 1/2 second.

5.6.1.6 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.

5.6.1.7 FrameRateInTrickMode

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. An OC-DMP MAY send 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

Note (Informative): The functionality provided by the FrameRateInTrickMode.ochn.org header has been superseded by the introduction of the FrameRateInTrickMode.dlna.org header [DLNA CR-214]. It is recommended that HNIMP OC-DMS and OC-DMP use the FrameRateInTrickMode headers as per Section 5.6 and Annex B.

5.6.1.8 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.

5.6.1.9 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].

5.6.2 Streaming Live Content to DLNA Clients

A DLNA client initiates streaming of live content by making an HTTP or RTP request using a URI value that maps to a <res> value referencing a content binary. Upon receipt of such a streaming request, the HNIMP OC-DMS SHALL map the request URI to the associated content binary as per the mapping logic in Section 5.6.2.1.

5.6.2.1 Request URI Mapping

1. If the incoming request is an RTP Play request, it is considered mapped if a <res> element is found whose URI matches the incoming request URI.
2. If the incoming request is an HTTP GET or HEAD request, it is considered mapped if one of the following returns a match:
    a. If registered, the HttpRequestResolutionHandler is invoked, and if the resolveHttpRequest method returns a URI, then the implementation uses this URI in place of the request URI in the next step.
    b. The implementation attempts to find a <res> URI that matches either the URI from Step-a or the incoming request URI if Step-a was not successful. If the match with <res> URI fails and if the <res> element contains the res@ocap:alternateURI property, the implementation attempts to match the request URI with the value of this property as per the matching rules specified in Annex C.1.4.1. The request URI is considered mapped if either a <res> URI or an alternateURI is matched as per their respective matching rules.

If the request URI is successfully mapped to a <res> value referencing a content binary, the HNIMP OC-DMS SHALL perform authorization checking as specified by the Network Authorization Handler in [HN-SEC].

NOTE (Informative): If the port number on which the OC-DMS listens for media streaming is also being used for other services (e.g., CDS), it may receive non-streaming related HTTP requests and may invoke the HttpRequestResolutionHandler for such requests. To avoid this, it is recommended that the OC-DMS use a separate port to listen for media streaming requests.

5.6.2.2 Tuner Resource Acquisition

If the request is authorized as per the Network Authorization Handler in [HN-SEC], then the OC-DMS HNIMP SHALL attempt to acquire a tuner in order to tune and stream content. If no tuner is available to tune to the requested

Service, the HNIMP SHALL resolve the conflict based on 19.2.1.1 of [OCAP]. If the conflict resolution requires invocation of a ResourceContentionHandler to resolve the contention, then the OC-DMS HNIMP SHALL pass a NetResourceUsage to represent the request. The HNIMP SHALL populate the NetResourceUsage object as follows:

- The NetResourceUsage.getInetAddress method SHALL return the InetAddress of the requesting client;
- The NetResourceUsage.getUsageType method SHALL return "Presentation";
- The NetResourceUsage.getAppId method SHALL return null;
- The NetResourceUsage.getResourceNames method SHALL return a single element array with an element value of "org.davic.net.tuning.NetworkInterfaceController";
- The NetResourceUsage.getOcapLocator method SHALL return an OcapLocator denoting the requested service.

When resolving the contention, the application priority associated with the NetResourceUsage SHALL be considered 0. If the contention cannot be resolved, then the OC-DMS HNIMP SHALL fail the streaming request with error 503 (service unavailable).

If a tuner is already tuned to the requested Service, then the HNIMP SHALL utilize the already-tuned tuner and any associated time-shift resources to honor the live streaming request. In the event of subsequent resource contention, the HNIMP SHALL represent the shared reservation of the tuner with a SharedResourceUsage containing a NetResourceUsage object as described above.

5.6.2.3 Tuning for Live Streaming

If a tuner is available that can stream the requested channel, then the OC-DMS HNIMP SHALL get the ChannelContentItem's Locator (returned by the ChannelContentItem.getChannelLocator method). The implementation SHALL take action based on the value of the returned Locator as follows:

- If the Locator is frequency based, attempt to tune to the channel,
- If the Locator is source ID based, perform a JavaTV Service lookup and:
    - If a Service is found and contains tuning parameters, attempt to tune to the channel.
    - If a Service is found but does not contain tuning parameters, determine if a SelectionProvider is registered for the "ocap://" scheme and supports the service represented by the ChannelContentItem, and if so, attempt to resolve tuning parameters as defined in Section 5.6.2.3.1.
    - If the Locator returned by getTuningLocator() is null, and a ServiceResolutionHandler is registered, resolve the tuning parameters by calling the ServiceResolutionHandler as defined in Section 5.6.2.3.2.
    - If ServiceResolutionHandler.resolveChannelItem returns True, attempt to tune to the locator returned by ChannelContentItem.getTuningLocator.
    - Otherwise fail the request.

See Figure 5–4.

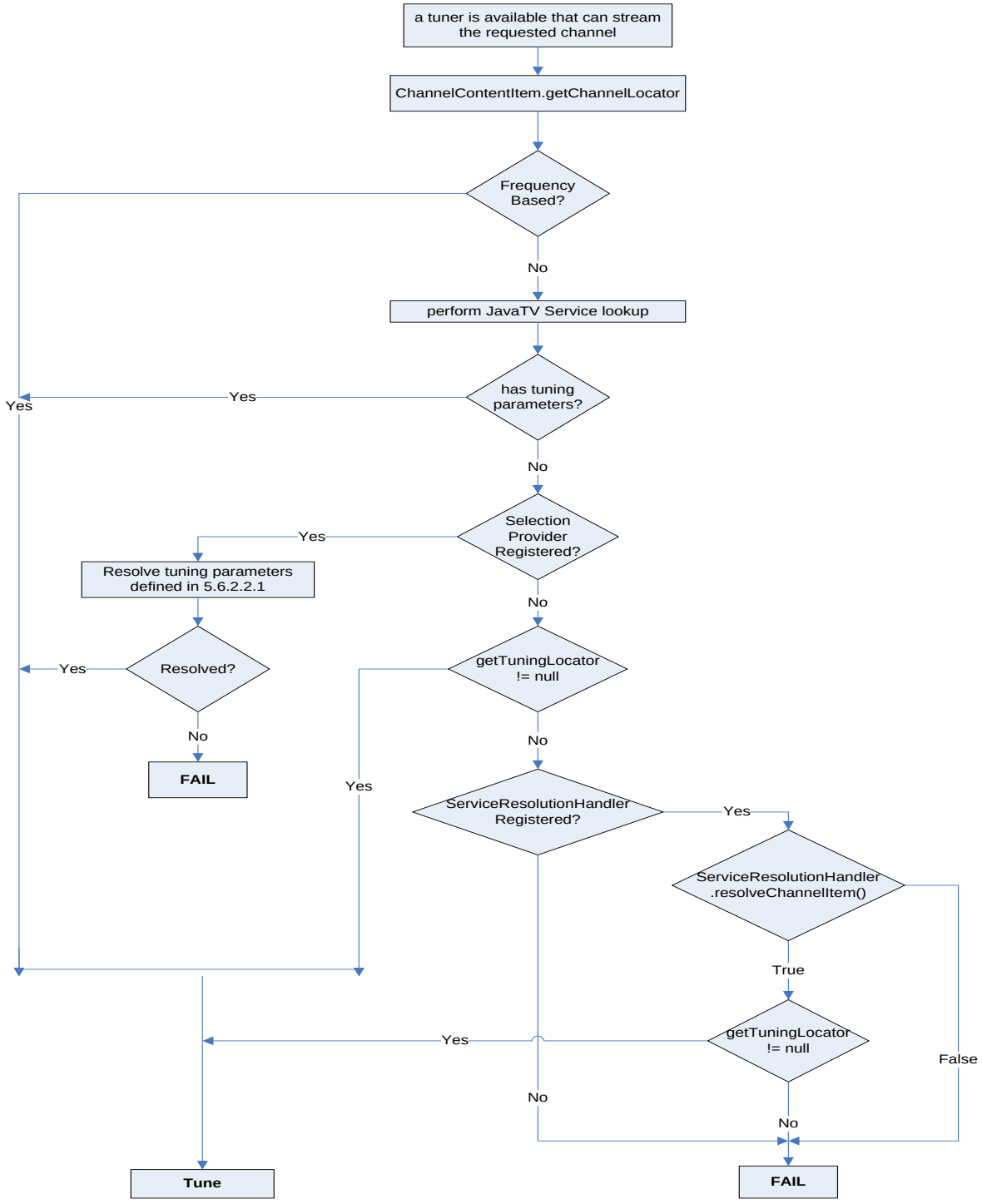

Figure 5–4 - Tuning for Live Streaming

5.6.2.3.1 Tuning Resolution with Service Provider

Section 10.2.2.2.5 of [OCAP] describes how the Service Provider Interface (SPI) defined in [DVB-MHP 1.1.3] Annex AN is complied with in tru2way for use with local Host SDV. The SPI can also be used for SDV during live streaming on a home network. When the OC-DMS HNIMP receives a streaming request containing a URI that refers to a ChannelContentItem that has a source ID based Locator, as returned from

ChannelContentItem.getChannelLocator, then the OC-DMS HNIMP SHALL determine if a SelectionProvider is registered that:

1. Is registered to handle the "ocap://" scheme, AND
2. Has indicated support for the Locator passed to the ContentEntryFactory.createChannelContentItem method via the SelectionProvider and SelectionProviderContext APIs.

In that case, the OC-DMS HNIMP SHALL handle live streaming activities for a ChannelContentItem when supported by a registered SelectionProvider as defined by [OCAP] section 10.2.2.2.5 and [DVB-MHP 1.1.3] Annex AN. That is, a JavaTV Service lookup is made based on the source ID Locator, and a SelectionSession is created and managed for each streaming activity to a SelectionProvider supported service.

The OC-DMS HNIMP SHALL fail the streaming request with error 503, Service Unavailable under any of the following conditions:

- If the Locator returned by SessionSelection.select does not work for tuning, or
- If the tune fails for any other reason.

When a SelectionProvider application uses the SPI to remove tuning information from a JavaTV Service being used for ChannelContentItem streaming, the implementation SHALL stop content streaming and close the connection.

5.6.2.3.2 Tuning Resolution with ServiceResolutionHandler

When the OC-DMS receives a streaming request containing a URI that refers to a ChannelContentItem that does not have a tuning Locator (that is, ChannelContentItem.getTuningLocator returns null), and if a ServiceResolutionHandler has been set by a call to ContentServerNetModule.setServiceResolutionHandler, then OC-DMS HNIMP SHALL notify the application by invoking ServiceResolutionHandler.resolveChannelItem, passing the untunable channel item as a parameter.

If the ServiceResolutionHandler application is able to establish an SDV session such that the channel becomes tunable, then it updates the ChannelContentItem's Locator (by calling ChannelContentItem.setTuningLocator) and the ServiceResolutionHandler.resolveChannelItem method returns true. If the ServiceResolutionHandler application is unable to establish an SDV session for the requested channel, then ServiceResolutionHandler.resolveChannelItem returns false.

Upon successful return from the ServiceResolutionHandler.resolveChannelItem method, the HNIMP SHALL call ChannelContentItem.getTuningLocator to obtain a Locator suitable for tuning the requested channel.

The OC-DMS HNIMP SHALL fail the streaming request with error 503, Service Unavailable under any of the following conditions:

- If the ServiceResolutionHandler.resolveChannelItem method returns false, or
- If there is no ServiceResolutionHandler and no SelectionProvider and the OC-DMS HNIMP is unable to obtain a Locator from the ChannelContentItem.getTuningLocator method, or
- If the OC-DMS HNIMP is able to obtain a Locator from the ChannelContentItem.getTuningLocator method, but is unable to tune the channel for any other reason.

For an ServiceResolutionHandler application to terminate an SDV session, it informs the HNIMP that the SDV channel is no longer available, by calling the ChannelContentItem.setTuningLocator method and passing null for the new Locator. If the ChannelContentItem.setTuningLocator method is called with a null argument, then the OC-DMS HNIMP SHALL terminate any active streaming sessions on this ChannelContentItem before returning from the ChannelContentItem.setTuningLocator method. Changing the tuning locator from a non-null value to a different non-null value also terminates an SDV session, but a new session is immediately available for streaming.

NOTE: An application may determine that there are no longer any streaming activities for an SDV channel based on notifyActivityStart / notifyActivityEnd callbacks. The application may choose to terminate an SDV session with the headend when it determines that there are no longer any streaming activities for an SDV channel; or at program boundaries; or at any other time based on any other application logic.

5.6.2.4 Live Streaming Media Transport

Upon successful tuning of a channel for a live streaming request, the OC-DMS HNIMP SHALL stream contents of the channel to the requesting client. The HNIMP SHALL protect the content as required by Section 5.10.

5.6.2.5 Live Streaming Trick Play Support

If the OC-DMS HNIMP is capable of supporting a TSB, then the OC-DMS HNIMP SHALL support the DLNA Limited Random Access Data Availability Model for the purposes of live streaming requests, as specified in Section 5.2.

5.6.2.6 Sequence Diagrams

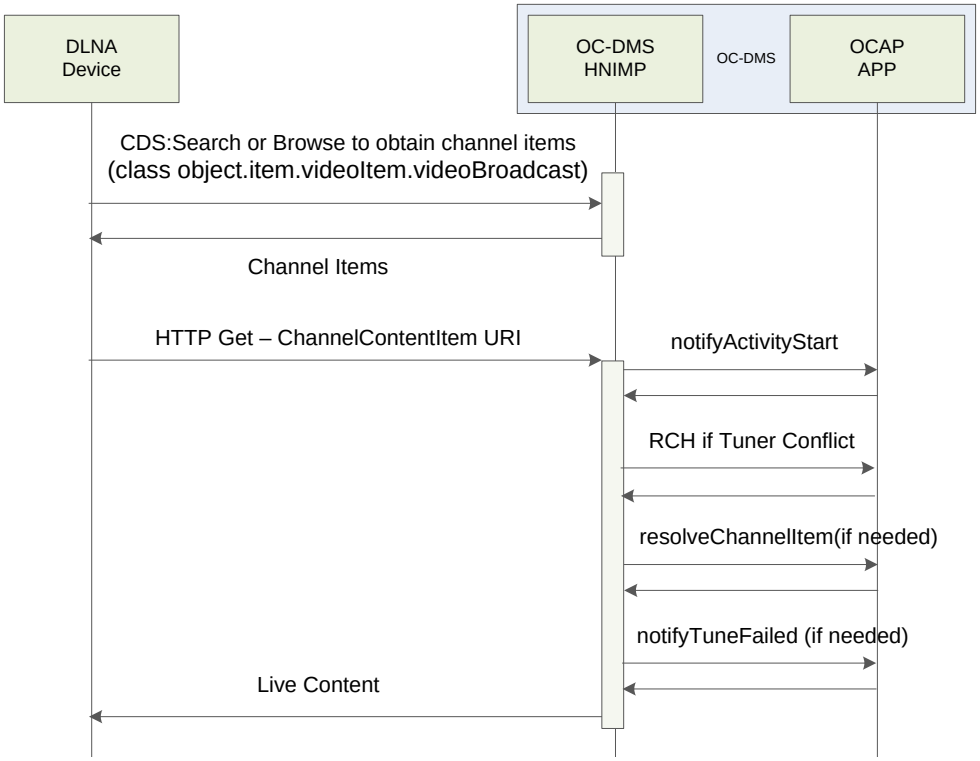

Figure 5–5 - Streaming Live Content to DLNA Device

5.6.3 Adaptive Streaming

When content is streamed using adaptive streaming over HTTP mechanism, the HNIMP OC-DMS SHALL respond to the HTTP Get request with a manifest file that conforms to the adaptive streaming mechanism (see Section 6.3.13.1). Based on the MIME type set in the res@protocolInfo property and the content of the manifest file received from the HNIMP OC-DMS, the client determines the following:

a. The adaptive streaming mechanism (e.g., MPEG-DASH) (Section 6.3.13.1)

b. Container Format of the content (MPEG-TS or MP4) (Section 6.3.13.1)

c. Required Link Protection (Section 6.3.13.2)

d. URI to the first fragment of the content

For MPEG-DASH, the client can determine any required DRM by parsing the manifest file received from the HNIMP OC-DMS.

The specific sequence of actions the HNIMP OC-DMP executes with the HNIMP OC-DMS depends on the specific combination of the above parameters that are applied to the content, e.g., if the content is within an MP4 container and requires DTCP-IP, the HNIMP OC-DMP needs to perform the PCP-UR subfunction of the DTCP-IP AKE (see [HN-SEC]).

This data determines the sequence of actions that the HNIMP OC-DMP performs and the sequence of messages the HNIMP OC-DMP exchanges with the HNIMP OC-DMS in order to complete the required DRM/Link Protection protocol successfully and initiate streaming of the selected content.

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 and DMS SHALL support content formats as specified in [OCAP]. This includes broadcast streaming, monomedia, and stored content formats. Additionally, an HNIMP DMS and DMP SHALL support the subset of the content formats defined in [DLNA vol 2] indicated in Table 5–2.

Table 5–2 - Mandatory and Optional Media Formats by Device Class

DLNA Profile	OC-DMS	OC-DMP	Informative Note
Image			
JPEG_SM	R	R	JPEG_SM is the DLNA HND required image format. The remaining formats are OCAP extensions to the DLNA requirement.
JPEG_MED	R	R	
JPEG_LRG	R	R	
JPEG_TN	R	R	
JPEG_SM_ICO	R	R	
JPEG_LRG_ICO	R	R	
PNG_TN	R	R	
PNG_SM_ICO	R	R	
PNG_LRG_ICO	R	R	
PNG_LRG	R	R	
GIF_LRG	R	R	
A/V			
MPEG_TS_NA_ISO	R	R	OC-DMS: Required per [HOST] OC-DMP: Required per [DLNA vol 5]
AVC_TS_NA_ISO	R	R	OC-DMS: Required per [HOST] OC-DMP: Required [DLNA vol 5]

DLNA Profile	OC-DMS	OC-DMP	Informative Note
AVC_TS_NA_T	O	R	OC-DMS: No requirement OC-DMP: Required [DLNA vol 5]
MPEG_TS_SD_NA_ISO	R	R	OC_DMS: Provides backward compatibility OC_DMP: Required per [DLNA vol 2] Table 6-1 and helps to maintains backward compatibility
MPEG_TS_HD_NA_ISO	R	R	OC_DMS: Maintains backward compatibility OC_DMP: Maintains backward compatibility
MPEG_TS_SD_NA_T	O	R	OC_DMP: Required per [DLNA vol 2] Table 6-1
MPEG_TS_HD_NA_T	O	R	Required for backward compatibility
MPEG_PS_NTSC	O	R	OC_DMP: Required per [DLNA vol 2] Table 6-1
MPEG_TS_SD_NA	O	R	OC_DMP: Required per [DLNA vol 2] Table 6-1
AVC_TS_NA_3DFC_ISO	R	O	OC_DMS: Required per [DLNA 3D MF]
MPEG_TS_NA_3DFC_ISO	R	O	OC_DMS: Required per [DLNA 3D MF]
Audio			
LPCM	R	R	DLNA HND requirement
AC3	R	R	OC_DMS: Required per [HOST] & [OCAP] OC_DMP: Required per [HOST] & [OCAP]
MP2_MPS	R	R	OC_DMS: Required per [HOST] & [OCAP] OC_DMP: Required per [HOST] & [OCAP]
MP3	R	R	OC_DMS: Required per [HOST] & [OCAP] OC_DMP: Required per [HOST] & [OCAP]

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).

An HNIMP OC-DMS SHALL support transformation of MPEG_TS_* (H.262) source content to AVC_TS_NA_ISO (H.264) based content.

In order to allow for appropriate audio channel selection, an HNIMP OC-DMS SHALL transform all audio streams and associated descriptor metadata into the transformed content resource when transforming MSO local content. Additionally, the HNIMP OC-DMS MAY create multiple audio stream representations from any audio streams within the source content.

When adaptive streaming over HTTP is supported, an HNIMP OC-DMS SHALL support MPEG-DASH as per [DLNA HTTP-AD].

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] allows for discovery and rendering of personal content as well as cable recorded services and the transport of cable-delivered broadcast content on the LAN. When cable recorded or live services are transported over a Home network, they SHALL be protected as defined in the [HN-SEC] specification.

When content is streamed using adaptive streaming over HTTP, the getProtectionType method on the ContentFormat interface returns the protection type to be used (see Section 6.3.13.2).

5.11 UPnP Interoperability

The HNIMP SHALL comply with UPnP interoperability guidelines as specified in [UPNP DA]. Figure 5–6 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.

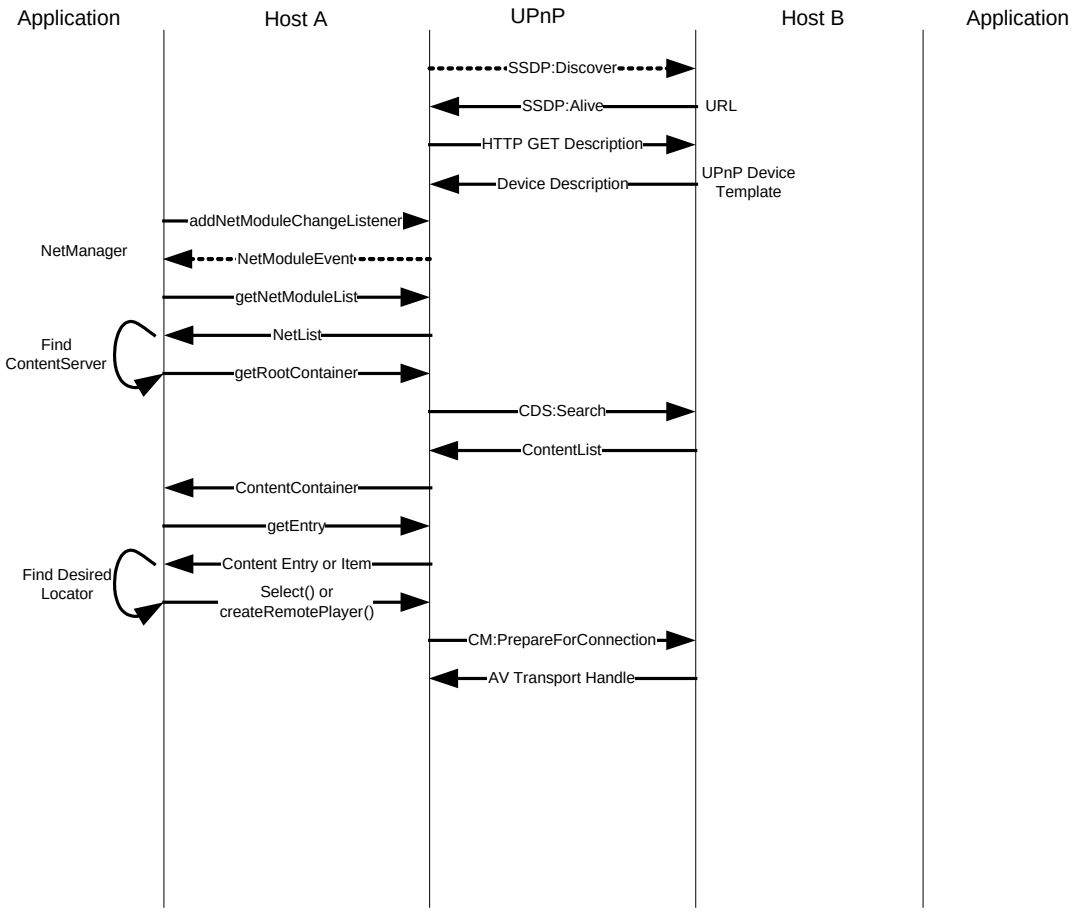

Figure 5–6 - Signal Flow Example

5.12 Parental Control

The [UPNP CDS] specification defines the upnp:rating property that can be used by clients to determine the parental control rating of a content item. An application may set the upnp:rating property using the MetadataNode API. An HNIMP OC-DMS SHALL NOT modify a upnp:rating property that has been set by an application.

The upnp:rating@type is used to define the rating dimension. The values that can be used in the United States include:

- "TV Parental Guideline Age-Based Rating for Entire Audience"
- "TV Parental Guideline Age-Based Rating for Children's Programs"
- "MPAA Rating"

The rating dimension values were taken from [CEA-766] section 5.1. Values for the Dialog Flag, Language Flag, Sexual Content Flag, Violence Flag, and Fantasy Violence Flag were not included because they are combined with other rating dimensions using upnp:rating property values; see Table 5–3.

The upnp:rating property can be used to represent values from multiple rating dimensions as defined in the Message column of Table 3 in [CEA-766]. Table 5–3 identifies the upnp:rating values that apply to each upnp:rating@type value:

Table 5–3 - Parental Control Ratings and Rating Dimensions

Rating Dimension (upnp:rating@type value)	Applicable Ratings (upnp:rating value)
"TV Parental Guideline Age-Based Rating for Children's Programs"	"TV–Y" "TV–Y7" "TV–Y7–FV"[1]
"TV Parental Guideline Age-Based Rating for Entire Audience"	"TV–None" "TV–G" "TV–PG" "TV–PG–D"[2] "TV–PG–L"[2] "TV–PG–S"[2] "TV–PG–V"[2] "TV–PG–D–L"[2] "TV–PG–D–S"[2] "TV–PG–D–V"[2] "TV–PG–L–S"[2] "TV–PG–L–V"[2] "TV–PG–S–V"[2] "TV–PG–D–L–S"[2] "TV–PG–D–L–V"[2] "TV–PG–D–S–V"[2] "TV–PG–L–S–V"[2] "TV–PG–D–L–S–V”[2] "TV–14"

Rating Dimension (upnp:rating@type value)	Applicable Ratings (upnp:rating value)
	"TV–14–D"[2] "TV–14–L"[2] "TV–14–S"[2] "TV–14–V"[2] "TV–14–D–L"[2] "TV–14–D–S"[2] "TV–14–D–V"[2] "TV–14–L–S"[2] "TV–14–L–V"[2] "TV–14–S–V"[2] "TV–14–D–L–S"[2] "TV–14–D–L–V"[2] "TV–14–D–S–V"[2] "TV–14–L-S–V"[2] "TV–14–D–L–S–V"[2] "TV–MA" "TV–MA–L"[2] "TV–MA–S"[2] "TV–MA–V"[2] "TV–MA–L–S"[2] "TV–MA–L–V"[2] "TV–MA–S–V"[2] "TV–MA–L–S–V"[2]
"MPAA Rating"	"MPAA–N/A" "MPAA–G" "MPAA–PG" "MPAA–PG13" "MPAA–R" "MPAA–NC–17" "MPAA–X" "MPAA–Not Rated"

Note 1: These rating values include the Fantasy Violence rating dimension.

Note 2: These rating values include one or more of the Dialog Flag, Language Flag, Sexual Content Flag, or Violence Flag rating dimensions.

5.13 View Primary Output Port (OPTIONAL)

Support for the View Primary Output Port (VPOP) feature is indicated by the ocHnNetConfigViewPrimaryOutputPort MIB object, as defined in [MIB-HN]. The HNIMP OC-DMS SHALL query the value of this MIB object at the beginning of normal operation (see [OCAP] "Boot Process"). If the value of the ocHnNetConfigViewPrimaryOutputPort MIB object is true, the HNIMP OC-DMS SHALL support the VPOP

feature, per the requirements listed below. If the value of the ocHnNetConfigViewPrimaryOutputPort MIB object is false or undefined, the HNIMP OC-DMS SHALL NOT support the VPOP feature. The HNIMP OC-DMS SHALL ignore any changes to the ocHnNetConfigViewPrimaryOutputPort MIB object after the start of normal operation.

5.13.1 View Primary Output Port (VPOP) Content Item

When the VPOP feature is supported, the HNIMP OC-DMS SHALL adhere to the following behaviors:

1. Create a content item in CDS (the "VPOP content item") representing the service currently presented to the default HScreen.
    a. Content streamed from a VPOP content item:
        - includes a presenting broadcast channel (including SDV and time-shifted content), completed and in-progress recordings, and VOD assets;
        - excludes presenting content streamed in from a home network interface;
        - may exclude analog services;
        - excludes graphics.
    b. The upnp:class of a VPOP content item is object.item.videoItem.vpop.
    c. The VPOP content item is published as an immediate child of the root container (i.e., @parentID equals 0).
    d. A <res> property is populated where the DLNA.ORG_PN field of the res@protocolInfo property matches the presenting content as defined in [DLNA vol 1]. The value of the DLNA.ORG_PN field is updated if the media format of the presenting content changes.
    e. The a-val and b-val parameters of the res@protocolInfo property DLNA.ORG_OP field will be set to '0', the DLNA.ORG_PS field will be omitted from the res@protocolInfo property, and the DMS will not support HTTP time or range headers for VPOP access. As a consequence, home network-based trick mode control of the VPOP content item will not be supported.
    f. The <res> URI value in a VPOP content item references the presenting content and is constant through changes to the presenting service.
    g. The ocap:accessPermissions property of the VPOP content item SHALL be set to read-only unless otherwise indicated by the ocHnNetConfigViewPrimaryOutputPortOrgID MIB object defined in [MIB-HN]. The HNIMP OC-DMS SHALL query the value of the ocHnNetConfigViewPrimaryOutputPortOrgID MIB object at the beginning of normal operation (see [OCAP] "Boot Process"). If the value of this MIB object is non-zero, then the HNIMP OC-DMS SHALL update the other_org field of the VPOP ocap:accessPermissions property to permit write access by applications whose organization ID match the value of the MIB object. Other fields in the VPOP ocap:accessPermissions property SHALL always be set to read-only (i.e., the owner_org-write, owner_app-write, and world-write flags SHALL be set to '0'). The HNIMP OC-DMS SHALL ignore any changes to the ocHnNetConfigViewPrimaryOutputPortOrgID MIB object after the start of normal operation.
    h. The dc:title property of a VPOP content item SHOULD be a descriptive value indicating its function.
2. When the URI of a VPOP content item is used to stream content using HTTP, the HNIMP OC-DMS will maintain the streaming session through changes to the presenting service so long as the media container format remains unchanged. The HNIMP OC-DMS will close the streaming session when the media container format of the content changes, but maintain the HTTP persistent connection per [DLNA vol 1] guidelines.
    a. If the HNIMP OC-DMS runs out of content during the service selection process, it MAY halt data transmission or transmit MPEG null packets until the selected content is available.

b. The service acquisition process may cause client underflow and subsequent client-specific behavior, such as black screen or freeze frame.

c. Streaming clients SHOULD continue playback of VPOP content through changes to the content format, including changes to resolution, frame rate and codec, as well as timestamp and PID discontinuities.

d. When content is streamed from a VPOP content item, DTCP encapsulation is always applied and content encryption is used as applicable to content CCI values. Copy freely content that does not have EPN asserted is not encrypted, but is sent in DTCP-IP Protected Content Packets. See [HN-SEC] for DTCP definition.

e. When broadcast content presenting from a TSB is streamed from a VPOP content item, then the content streamed represents the presentation point in the TSB.

f. When content is streamed from a VPOP content item, it is not altered to match video scaling of the presentation.

g. Per the requirements of [DLNA vol 1] section 7.4 "Media Transport", the HNIMP OC-DMS SHALL maintain media format compliance of VPOP streamed content during trick-mode playback of content presented to the default HScreen. The HNIMP OC-DMS MAY exclude user data and audio from the VPOP streamed content during trick mode playback. The HNIMP OC-DMS MAY halt data transmission or transmit MPEG null packets until normal playback rate is resumed.

h. Only a single streaming session associated with a VPOP content item is permitted at a time. If an HNIMP OC-DMS has an in-progress VPOP session with a client and the client sends a request for VPOP content, the HNIMP OC-DMS SHALL close the in-progress session and process the request. If an HNIMP OC-DMS has an in-progress VPOP session and a request for VPOP content comes from a client that is not part of the in-progress session, then the HNIMP OC-DMS SHALL respond to the second streaming request with HTTP error 503 (service temporarily overloaded).

i. The HNIMP OC-DMS SHALL deny streaming requests originating from the same device hosting the HNIMP OC-DMS.

3. Support the View Primary Output Port Service; see Section 5.13.2.

5.13.2 View Primary Output Port (VPOP) Service

When the VPOP feature is supported, the View Primary Output Port service SHALL be added to the OCAP Media Server, as defined below. This service SHALL have the service type “urn:cablelabs-com:service:ViewPrimaryOutputPort:1”.

5.13.2.1 State Variables

5.13.2.1.1 A_ARG_TYPE_ConnectionID

This state variable provides the ConnectionID value used as an IN argument to several actions of this service. This state variable is not evented. In general, the value of the ConnectionID parameter must match the ConnectionID of an in-progress VPOP content item for the action to succeed.

5.13.2.1.2 X_ARG_TYPE_PowerStatus

This state variable provides the power status value returned as an OUT argument by the PowerStatus() action. This state variable is not evented.

5.13.2.1.3 X_ARG_TYPE_TuneParameters

This state variable provides the channel values used as an IN argument to the Tune() action. This state variable is not evented.

The TuneParameters argument is formatted according to the following BNF:

<major_channel_number>[“,”<minor_channel_number>]

major_channel_number = decimal_string

minor_channel_number = decimal_string

decimal_string = 1*digit

digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

5.13.2.2 Actions

5.13.2.2.1 AudioMute()

This action allows a rendering endpoint or control point to pass a "mute audio" remote control command to an HNIMP OC-DMS. Upon receiving a call to this action, the HNIMP OC-DMS SHALL pass the command to the OCAP Host device for processing. If the OCAP Host device is in an unmuted state, it SHALL take action as if the "Mute Volume" key had been pressed on its remote control; otherwise, it SHALL do nothing. See [OCAP]"Input Events" for more information.

If the value of the ConnectionID parameter does not match the ConnectionID of an in-progress VPOP content item streaming session, then the HNIMP OC-DMS SHALL respond with UPnP error code 606 Action Not Authorized.

5.13.2.2.1.1 Arguments

Argument	Direction	Related State Variable
ConnectionID	*IN*	*A_ARG_TYPE_ConnectionID; see 5.13.2.1.1*

5.13.2.2.1.2 Dependency on State

None.

5.13.2.2.1.3 Effect on State

None.

5.13.2.2.1.4 Errors

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.

5.13.2.2.2 AudioRestore()

This action allows a rendering endpoint or control point to pass a "restore audio" remote control command to an HNIMP OC-DMS. Upon receiving a call to this action, the HNIMP OC-DMS SHALL pass the command to the OCAP Host device for processing. If the OCAP Host device is in a muted state, it SHALL take action as if the "Mute Volume" key had been pressed on its remote control; otherwise, it SHALL do nothing. See [OCAP] "Input Events" for more information.

If the value of the ConnectionID parameter does not match the ConnectionID of an in-progress VPOP content item streaming session, then the HNIMP OC-DMS SHALL respond with UPnP error code 606 Action Not Authorized.

5.13.2.2.2.1 Arguments

Argument	Direction	Related State Variable
ConnectionID	*IN*	*A_ARG_TYPE_ConnectionID; see 5.13.2.1.1*

5.13.2.2.2.2 Dependency on State

None.

5.13.2.2.2.3 Effect on State

None.

5.13.2.2.2.4 Errors

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.

5.13.2.2.3 PowerOn()

This action allows a rendering endpoint or control point to pass a "power on" remote control command to an HNIMP OC-DMS. Upon receiving a call to this action, the HNIMP OC-DMS SHALL pass the command to the OCAP Host device for processing. If the OCAP Host device is in standby power mode, it SHALL take action as if the "Power On/Off" key had been pressed on its remote control; otherwise, it SHALL do nothing. See [OCAP] "Input Events" for more information.

5.13.2.2.3.1 Arguments

None.

5.13.2.2.3.2 Dependency on State

None.

5.13.2.2.3.3 Effect on State

None.

5.13.2.2.3.4 Errors

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.

5.13.2.2.4 PowerOff()

This action allows a rendering endpoint or control point to pass a "power off" remote control command to an HNIMP OC-DMS. Upon receiving a call to this action, the HNIMP OC-DMS SHALL pass the command to the OCAP Host device for processing. If the OCAP Host device is in full power mode, it SHALL take action as if the "Power On/Off" key had been pressed on its remote control; otherwise, it SHALL do nothing. See [OCAP] "Input Events" for more information.

If the value of the ConnectionID parameter does not match the ConnectionID of an in-progress VPOP content item streaming session, then the HNIMP OC-DMS SHALL respond with UPnP error code 606 Action Not Authorized.

5.13.2.2.4.1 Arguments

Argument	Direction	Related State Variable
ConnectionID	*IN*	*A_ARG_TYPE_ConnectionID; see 5.13.2.1.1*

5.13.2.2.4.2 Dependency on State

None.

5.13.2.2.4.3 Effect on State

None.

5.13.2.2.4.4 Errors

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.

5.13.2.2.5 PowerStatus()

This action allows a rendering endpoint or control point to query the power status of an HNIMP OC-DMS. This action will report the current power status of the OCAP Host device, either FULL POWER or STANDBY.

5.13.2.2.5.1 Arguments

Argument	Direction	Related State Variable
PowerStatus	*OUT*	*X_ARG_TYPE_PowerStatus; see 5.13.2.1.2*

5.13.2.2.5.2 Dependency on State

None.

5.13.2.2.5.3 Effect on State

None.

5.13.2.2.5.4 Errors

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.

5.13.2.2.6 Tune()

This action allows a rendering endpoint or control point to pass a "tune" remote control command to an HNIMP OC-DMS. Upon receiving a call to this action, the HNIMP OC-DMS SHALL pass the command and its parameters to the OCAP Host device. The OCAP Host device SHALL then generate the following key events, in order:

- VK_0 through VK_9, corresponding to the major_channel_number component of the TuneParameters argument,
- if the minor_channel_number component is present in the TuneParameters argument: VK_PERIOD followed by VK_0 through VK_9 corresponding to minor_channel_number,
- VK_ENTER.

See [OCAP] "Input Events" for more information.

If the value of the ConnectionID parameter does not match the ConnectionID of an in-progress VPOP content item streaming session, then the HNIMP OC-DMS SHALL respond with UPnP error code 606 Action Not Authorized. If the value of the TuneParameters argument does not match the required syntax, the HNIMP OC-DMS SHALL respond with UPnP error code 600 Argument Value Invalid.

5.13.2.2.6.1 Arguments

Argument	Direction	Related State Variable
ConnectionID	*IN*	*A_ARG_TYPE_ConnectionID; see 5.13.2.1.1*
TuneParameters	*IN*	*X_ARG_TYPE_TuneParameters; see 5.13.2.1.3*

5.13.2.2.6.2 Dependency on State

None.

5.13.2.2.6.3 Effect on State

None.

5.13.2.2.6.4 Errors

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.

5.14 UPnP Diagnostics

An HNIMP OC-DMS SHALL support the +DIAGE+ capability defined in [DLNA Diag].

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] requirements 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. Any UPnP Remote UI device, if supported, that is part of the +RUIHSRC+ capability SHALL also be included as a sub-device of the root device. Any UPnP Device Management service that is part of the +DIAGE+ capability SHALL be included as a service of the root device.

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 Inet4Address 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 getActionStatus
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

All of the getDeviceList methods 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 getNetModuleList method

All of the getNetModuleList methods SHALL return a list of org.ocap.hn.NetModule objects created during discovery as defined in Section 6.1. NetModules in the list SHALL be ordered by root device, then sub-devices within the root. Local devices SHALL appear in the list first.

6.2.7.4 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).

To obtain the sample frequency value using org.ocap.hn.content.ContentResource.getResourceProperty(String key), the key parameter value SHALL be “res@sampleFrequency”. ContentResource.getResourceProperty(String key) SHALL return a java.lang.Integer wrapping the same integer value returned by the AudioResource.getSampleFrequency() method. If the sample frequency is unknown, then a java.lang.Integer representing a value of -1 SHALL be returned.

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.

To obtain the number of audio channels using org.ocap.hn.content.ContentResource.getResourceProperty(String key), the key parameter value SHALL be "res@nrAudioChannels". ContentResource.getResourceProperty(String key) SHALL return a java.lang.Integer wrapping the same integer value returned by the AudioResource.getNumberOfChannels() method. If the number of channels is unknown, then a java.lang.Integer representing a value of -1 SHALL be returned.

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.

To obtain the number of bits per sample using org.ocap.hn.content.ContentResource.getResourceProperty(String key), the key parameter value SHALL be "res@bitsPerSample". ContentResource.getResourceProperty(String key) SHALL return a java.lang.Integer wrapping the same integer value returned by the AudioResource.getBitsPerSample() method. If the bits per sample is unknown, then a java.lang.Integer representing a value of -1 SHALL be returned.

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.

To obtain the array of Strings returned by getLanguages() using org.ocap.hn.content.ContentResource.getResourceProperty(String key), the key parameter value SHALL be "dc:language". ContentResource.getResourceProperty(String key) SHALL return an array of java.lang.String identical to the array of Strings returned by the AudioResource.getLanguages() method. If the languages for the resource are unknown, then a zero length array SHALL be returned.

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.

If this container has class object.container.channelGroup (or a subclass), and the parameter is NOT an instance of ChannelContentItem, then the HNIMP Shall return false.

Otherwise, if the ContentEntry parameter has a parentID property, the HNIMP SHALL remove the CDS entry with an @id property matching the value returned from the ContentEntry.getID method from the CDS content container whose @id property matches the ID contained in the parentID property, and SHALL add the CDS entry to the CDS content container matching the parentID 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 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 RecordingContentItem is not an instance of OcapRecordingRequest, then this method SHALL return false. 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.

When the parameter is an instance of ChannelContentItem, the HNIMP SHALL create a CDS content item and add it to the CDS container corresponding to this ContentContainer (see Section 6.3.2.5). The HNIMP SHALL publish a res property for each associated ContentResource. Each res@protocolInfo property SHALL indicate the supported operations for the content binary indicated by the URI of the corresponding res property (see Section 5.2). The URI of the res property is implementation-dependent. The HNIMP SHALL include a TSPEC as defined in Annex D.

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. The OC-DMS HNIMP SHALL add the entries into the container in their order in the array argument (this is important for ordering of ChannelContentItems; see [DLNA vol 1] B.2.1).

6.3.2.3 createContentContainer method

The HNIMP SHALL create a CDS container corresponding to the newly-created ContentContainer and add it to the CDS at a location where this ContentContainer is the parent. 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 createChannelGroupContainer method

When this method is called, the HNIMP SHALL create a content container with class object.container.channelGroup as a child of this ContentContainer. The OC-DMS HNIMP SHALL set the channel group container's dlna:containerType property to the value "Tuner_1_0". The OC-DMS HNIMP SHALL set the channel group container's dc:title property to the value of the name argument. If this ContentContainer's isLocal method returns false, then this method SHALL return false.

The OC-DMS HNIMP SHALL include the objectID of this ChannelGroup in the Tuner Feature returned by the CDS GetFeatureList action.

6.3.2.5 createContentItem method

The OC-DMS HNIMP SHALL return false if this container is an instance of object.container.ChannelGroup. The HNIMP SHALL create a CDS item corresponding to the newly-created ContentItem and add it to CDS at a location where this ContentContainer is the parent. 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.

6.3.2.6 delete method

When the delete method is invoked, the HNIMP SHALL remove the corresponding local CDS container with the @id property that matches the ObjectID as return value of the getID method of this ContentContainer, if and only if that container does not contain any other CDS entries. Upon successful removal from the CDS, the method returns true. If the local CDS container contains other CDS entries, then the HNIMP SHALL take no action and the method returns false.

6.3.2.7 deleteContents method

When the deleteContents method is invoked, the HNIMP SHALL remove all direct child CDS items from the local CDS container. Direct child items are identified as those with an @parentID property that matches the return value of the getID method of this ContentContainer, i.e., the @id property of the CDS container corresponding to this ContentContainer. The HNIMP SHALL use the deleteEntry method of each direct child CDS item for the removal.

6.3.2.8 deleteEntry method

When the deleteEntry method is invoked, the HNIMP SHALL remove the corresponding local CDS container with the @id property matching the return value of the getID method of this ContentContainer. If the CDS container with the same @id contains CDS entries, those entries will also be removed from the CDS recursively using the deleteEntry method mapping as defined for each entry's corresponding ContentEntry type.

6.3.2.9 getContainerClass method

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

6.3.2.10 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 the RecordingContentItem 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. Finally, the HNIMP also removes the ContentEntry parameter from the corresponding NetRecordingEntry.

5. When the parameter is a NetRecordingEntry that contains a upnp:srsRecordScheduleID and contains no RecordingContentItemItem 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, th 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.

When this method is used to remove a CDS container, the HNIMP SHALL recursively remove all child CDS entries of that container and generate corresponding CDS changed events. Moderated CDS events sent out at implementation-specific intervals during the removal SHALL have the <stUpdate> element set to "1" for items that are part of the removal. When the removal is complete, the CDS event containing the last of the events corresponding to the removal SHALL include an <stDone> element for the subtree root of the removal.

Removal of a CDS content item SHALL have no effect on in-progress streaming session for that content item, i.e., any in-progress streaming session for a CDS content item will remain unaffected even after a CDS content item is removed.

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].

6.3.2.11 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 deleteEntry method

When the deleteEntry method is invoked, the HNIMP SHALL remove the local CDS entry with the @id property that matches the return value of the getID method of this ContentEntry.

6.3.3.2 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 ContentEntrys.

6.3.3.3 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 ContentEntrys that are not RecordingContentItems unless directed to do so by an application.

The HNIMP SHALL ensure that the res@size property for a locally hosted RecordingContentItem that represents a recording in progress is set at the conclusion of the recording.

6.3.3.4 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 ContentEntrys that are not RecordingContentItems unless directed to do so by an application.

6.3.3.5 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 ContentEntrys that are not RecordingContentItems unless directed to do so by an application.

6.3.3.6 getLocation method

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

6.3.3.7 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.

6.3.3.8 getParentID method

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

6.3.3.9 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 ChannelContentItem Interface

A ChannelContentItem represents a channel that is exposed to clients in a ChannelGroup container. ChannelContentItems are created via the ContentEntryFactory.createChannelContentItem method.

6.3.5 ContentEntryFactory class

6.3.5.1 createChannelContentItem method

When this method is called, the OC-DMS HNIMP SHALL create an instance of the ChannelContentItem interface. The OC-DMS HNIMP SHALL create a ContentResource object corresponding to each supported protocol (such as HTTP, RTP).

The OC-DMS HNIMP SHALL set channel content item's @restricted property to "1".

The OC-DMS HNIMP SHALL create a MetadataNode in the ChannelContentItem with the following properties:

- upnp:class populated from the channelType argument (required). For compliance with [UPNP CDS], the upnp:class SHALL be set to "object.item.videoItem.videoBroadcast" when "object.item.videoItem.videoBroadcast.vod" is provided as the channelType.
- upnp:channelID, populated from channelNumber argument per the requirements of [UPNP CDS] and [UPNP SRS] (if non-null)

- upnp:channelID@type, attribute of the upnp:channelID property; the channelID must be of type DIGITAL or ANALOG (required if there is a channelID property)
- upnp:channelNr, populated from the value of the upnp:channelID property, per the requirements of [UPNP CDS] (if non-null)
- dc:title: populated from the channelTitle argument (required)
- upnp:channelName populated from the channelName argument (required)
- ocap:appID populated from the application identifier of the application that called this method
- ocap:accessPermissions populated from the permissions parameter

The OC-DMS HNIMP SHALL configure every res property in a channel content item for DTCP-IP as defined in [HN-SEC].

6.3.6 ContentItem Interface

6.3.6.1 deleteEntry method

When ContentItem.deleteEntry method is invoked, the HNIMP SHALL perform the same steps as identified in the mapping of ContentEntry.deleteEntry. In addition, the HINMP SHALL remove from storage all binaries represented by <res> blocks in the CDS item mapping this ContentItem.

6.3.7 ContentResource interface

An HNIMP OC-DMS SHALL include one and only one ContentResource in a ContentItem object for each <res> element representing non-transformed (non-converted) content. An HNIMP OC-DMS SHALL only expose transformed representations in content items when specifically enabled via any of the transformation setter methods in the TransformationManager.

6.3.7.1 getContentLocator method

This method returns a Locator that can be used on the OC-DMP for service selection, in order to initiate streaming.

If the ContentResource is associated with a RecordingContentItem, then the getLocator method returns a Locator that maps to the required ocap:contentURI property.

If the ContentResource is associated with a ChannelContentItem, then the getLocator method returns a Locator based on the URI value of the res property.

6.3.7.2 getProtocol method

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

6.3.7.3 getNetwork method

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

6.3.7.4 getContentFormat

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

6.3.7.5 get ResourceProperty method

The getResourceProperty method returns properties of the resource. Subinterfaces of ContentResource provide access to a set of defined properties that are specific to each subinterface. The subinterface is responsible for identifying the key parameter value to getResourceProperty() for each resource property.

The getResourceProperty method also allows for custom or new properties. The getResourceProperty method SHALL return a java.lang.String when a property is not mapped to a specific data type and is present in the resource metadata. Null SHALL be returned if the key parameter does not match any property.

6.3.7.6 getTransformedContentFormats method

For content resources acquired from local content items, the getTransformedContentFormats method SHALL return an array of available transformed ContentFormat instances. For HTTP Adaptive content, there SHALL be one element representing each Representation in the MPD. For all other formats, the method SHALL return a single instance representing the transformed content binary format.

NOTE: For HTTP Adaptive content, there may be multiple ContentFormat instances per AdaptationSet that have the same content protection parameters.

6.3.8 IOStatus interface

6.3.8.1 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.

6.3.9 MetadataNode class

A MetadataNode has a collection of metadata entries, each of which is a key/value pair where the key and value map to the name and value, respectively, of a CDS property. CDS properties and their names and values are described in [UPNP CDS] section 2.2.20. The rest of this section defines the mapping from keys and values to CDS property names and values.

The key parameter for various methods in the MetadataNode class can include the OCAP application default name space (i.e., "ocapApp"), a vendor-defined name space, a standardized name space (Section 5.5.2), (e.g., "dc", "upnp", "ocap"), or no name space, which implies the OCAP application default namespace (i.e., "ocapApp"). As specified in Section 5.5.2, the HNIMP uses "ocap" namespace for properties defined in Annex C.

NOTE (informative): DIDL-Lite is the default namespace in DLNA, so no prefix is required for DIDL-Lite defined properties in DLNA. See guideline 7.3.25.2 of [DLNA vol 1]. However, for MetadataNode objects, "ocapApp" is the default namespace, so applications need to explicitly specify "didl-lite" as the namespace prefix when dealing with DIDL-Lite properties in a MetadataNode.

A key maps to a CDS property name in the following manner:

1. If the MetadataNode does not have a parent MetadataNode, the CDS property name is the key.

2. If the MetadataNode has a parent MetadataNode, the CDS property name is the CDS property name to which the key of the MetadataNode with respect to its parent MetadataNode maps, followed by “::”, followed by the key.

NOTE (informative): For example, in an unparented MetadataNode, suppose the key in a key/value pair is “upnp:foreignMetadata” and the value is a reference to another MetadataNode. Then within the first MetadataNode

the key “upnp:foreignMetadata” maps to the CDS property name “upnp:foreignMetadata”, and within the second MetadataNode the key “fmBody” maps to the CDS property name “upnp:foreignMetadata::fmBody”.

NOTE (informative): A CDS property is dependent or independent, depending on whether or not, respectively, its name contains an “@” symbol. A dependent CDS property is associated with either zero or one independent CDS property.

A value maps to a CDS property value in the following manner:

[1] If the key in a key/value pair maps to the name of an independent CDS property:

[1.1] If the value is a reference to a nonarray object, then the CDS property is single-valued, and the value of its single occurrence is determined as follows:

[1.1.1] If the value is a reference to an instance of java.lang.String, the CDS property value is the string value.

[1.1.2] If the value is a reference to an instance of java.lang.Integer and the key is not ocap:scheduledDuration, the CDS property value is the integer value as computed by the java.lang.Integer.toString() method. If the value is a reference to an instance of java.lang.Integer and the key is ocap:scheduledDuration, the CDS property value is as defined in Section C.1.1.3.

[1.1.3] If the value is a reference to an instance of java.lang.Long, the CDS property value is the long value as computed by the java.lang.Long.toString() method.

[1.1.4] If the value is a reference to an instance of java.lang.Boolean, the CDS property value is the Boolean value as computed by the java.lang.Boolean.toString() method.

[1.1.5] If the value is a reference to an instance of java.util.Date, the CDS property value is the date/time value in the "yyyy-mm-ddThh:mm:ss" format defined by [ISO 8601].

[1.1.6] If the value is a reference to a fully serializable instance of another class, the CDS property value is as defined in Section 6.3.9.1.1.

[1.1.7] If the value is a reference to an instance of a class that extends MetadataNode, the CDS property value is as defined in Section 6.3.9.1.1.

[1.1.8] If the value is a reference to an instance of org.dvb.application.AppID, the CDS property value is as defined in Section C.1.1.10.

[1.1.9] If the value is a reference to an instance of org.ocap.storage.ExtendedFileAccessPermissions, the CDS property value is as defined in Section C.1.1.8.

[1.2] If the value is a reference to an array object, then the CDS property is multi-valued, and the values of its multiple occurrences are determined by the values of the elements of the array object, in order of increasing subscript *I*, as follows:

[1.2.1] If the *i*th value is null, the *i*th CDS property value is the empty string.

[1.2.2] If the *i*th value is not null, the *i*th CDS property value is as defined in clause [1.1].

[2] If the key in a key/value pair maps to the name of a dependent CDS property that is associated with an independent CDS property:

[2.1] If the value is a reference to a java.lang.String object, then the associated independent CDS property is single-valued, the dependent CDS property occurs within the independent CDS property, and its value is the string value.

[2.2] If the value is a reference to a java.lang.String[] object, then the associated independent CDS property is multi-valued, and the nature of the dependent CDS property within the multiple occurrences of the independent CDS property is determined by the values of the elements of the java.lang.String[] object, in order of increasing subscript *i*, as follows:

[2.2.1] If the *i*th value is null, the *i*th dependent CDS property does not occur.

[2.2.2] If the *i*th value is not null, the *i*th dependent CDS property occurs, and its value is the string value.

[3] If the key in a key/value pair maps to the name of a dependent CDS property that is not associated with an independent CDS property, the CDS property value is the string value.

Immediately before and immediately after invocation of each public MetadataNode method, the following invariant SHALL hold: for each key/value pair k_d/v_d where the key maps to the name of a dependent CDS property with an associated independent CDS property, there shall be a key/value pair k_i/v_i where the key maps to the name of the associated independent CDS property, and exactly one of the following statements SHALL be true:

- v_d is a reference to a java.lang.String object and v_i is a reference to a nonarray object.
- v_d is a reference to a java.lang.String[] object and v_i is a reference to an array object of the same (nonzero) length.

NOTE (informative): For example, if an addMetadata method is called to associate a single URI with the key "upnp:author@role", then a single value must thereby be associated with the key "upnp:author" as well; but if an addMetadata method is called to associate an array of two URIs with the key "upnp:author@role", then an array of two values must thereby be associated with the key "upnp:author" as well.

When the MetadataNode has an associated CDS content entry, the HNIMP SHALL maintain an association of that CDS content entry's properties to MetadataNode fields in an implementation-specific fashion for the lifetime of the MetadataNode object.

6.3.9.1 addMetadata method

The format of the key argument is expected to be in the format defined in [UPNP CDS] section 2.2.20, with the exception that child (nested) properties are identified by their simple names, not their fully qualified names.

When one of these method is called, the HNIMP SHALL adhere to the following steps in order:

1. Determine the namespace of the property to be added, modified, or deleted. See Section 6.3.9 for more details. If the namespace 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 argument to existing property names in the pre-determined namespace, 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 property addition, modification, or deletion as follows:

   [3.1] If the value argument is neither null nor a reference to a zero-length array object and the key argument was not found in the property namespace, then add a new metadata entry to the MetadataNode and a new property to the associated CDS content entry (if there is one) using the key and value arguments.

[3.2] If the value argument is neither null nor a reference to a zero-length array object and the key argument was found in the property namespace, then change the corresponding metadata entry value and the corresponding property value (if there is one) using the value argument.

[3.3] If the value argument is null or a reference to a zero-length array object, then delete the corresponding metadata entry value (if there is one) and the corresponding property value (if there is one).

4. Establish the invariant of Section 6.3.9 as follows:

[4.1] If the key argument maps to the name of an independent CDS property with associated dependent CDS properties:

[4.1.1] If the property change was an addition, do nothing.

[4.1.2] If the property change was a modification from a reference to object A to a reference to object B, do the appropriate one of the following for each associated dependent CDS property p:

[4.1.2.1] If A and B are both nonarray objects, do nothing.

[4.1.2.2] If A and B are both array objects of the same length, do nothing.

[4.1.2.3] If A is a nonarray object and B is an array object of length n, process a property modification for p that changes its String value v to the String[] value w, computed as if by the statement (w = new String[n])[0] = v;. (NOTE (informative): In other words, convert its value to an array by extending it "on the right" with nulls.)

[4.1.2.4] If A is an array object and B is a nonarray object, compute the String value v from p's String[] value w as if by the statement v = w[0];; then, if v is null, process a property deletion for p, otherwise process a property modification for p that changes its value to v. (NOTE (informative): In other words, truncate its value "on the right".)

[4.1.2.5] If A and B are both array objects and the length of A is less than the length of B, process a property modification for p that null-extends its String[] value to the length of B.

[4.1.2.6] If A and B are both array objects and the length of A is greater than the length of B, process a property modification for p that truncates its String[] value to the length of B.

[4.1.3] If the property change was a deletion, process property deletions for all associated dependent CDS properties.

[4.2] If the key argument maps to the name of a dependent CDS property associated with an independent CDS property p:

[4.2.1] If the property change was an addition of a reference to object B, do the appropriate one of the following for p:

[4.2.1.1] If there is not yet a key/value pair where the key maps to the name of p, process a property addition for p that adds its value defined as follows:

[4.2.1.1.1] If B is a String object, the value is the empty string.

[4.2.1.1.2] If B is a String[] object of length n, the value is the String[] value w, computed as if by the statement `(w = new String[n])[0] =`

`"";`. (NOTE (informative): In other words, it gets an array value that is an empty string extended "on the right" with nulls.)

[4.2.1.2] If there is already a key/value pair where the key maps to the name of p and the value is a reference to object A, do the appropriate one of the following for p:

[4.2.1.2.1] If A and B are both nonarray objects, do nothing.

[4.2.1.2.2] If A and B are both array objects of the same length, do nothing.

[4.2.1.2.3] If A is a nonarray object and B is a String[] object of length n, process a property modification for p that changes its T value v to the T[] value w, computed as if by the statement `(w = new T[n])[0] = v;`. (NOTE (informative): In other words, convert its value to an array by extending it "on the right" with nulls.) Propagate the property change for p to other dependent CDS properties, as described in clause [4.1.2.3].

[4.2.1.2.4] If A is an array object and B is a String object, compute the T value v from p's T[] value w as if by the statement `v = w[0];`; then, if v is null, process a property deletion for p, otherwise process a property modification for p that changes its value to v. (NOTE (informative): In other words, truncate its value "on the right".) Propagate the property change for p to other dependent CDS properties, as described in clause [4.1.3] or [4.1.2.4], respectively.

[4.2.1.2.5] If A and B are both array objects and the length of A is less than the length of B, process a property modification for p that null-extends its T[] value to the length of B. Propagate the property change for p to other dependent CDS properties, as described in clause [4.1.2.5].

[4.2.1.2.6] If A and B are both array objects and the length of A is greater than the length of B, process a property modification for p that truncates its T[] value to the length of B. Propagate the property change for p to other dependent CDS properties, as described in clause [4.1.2.6].

[4.2.2] If the property change was a modification from a reference to object A to a reference to object B, do the appropriate one of the following for p:

[4.2.2.1] If A and B are both String objects, do nothing.

[4.2.2.2] If A and B are both String[] objects of the same length, do nothing.

[4.2.2.3] If A is a String object and B is a String[] object of length n, process a property modification for p that changes its T value v to the T[] value w, computed as if by the statement `(w = new T[n])[0] = v;`. (NOTE (informative): In other words, convert its value to an array by extending it "on the right" with nulls.) Propagate the property change for p to other dependent CDS properties, as described in clause [4.1.2.3].

[4.2.2.4] If A is a String[] object and B is a String object, compute the T value v from p's T[] value w as if by the statement `v = w[0];`; then, if v is null, process a property deletion for p, otherwise process a property modification for p that changes its value to v. (NOTE (informative): In other words, truncate its value "on the right".)

Propagate the property change for p to other dependent CDS properties, as described in clause [4.1.3] or [4.1.2.4], respectively.

[4.2.2.5] If A and B are both String[] objects and the length of A is less than the length of B, process a property modification for p that null-extends its T[] value to the length of B. Propagate the property change for p to other dependent CDS properties, as described in clause [4.1.2.5].

[4.2.2.6] If A and B are both String[] objects and the length of A is greater than the length of B, process a property modification for p that truncates its T[] value to the length of B. Propagate the property change for p to other dependent CDS properties, as described in clause [4.1.2.6].

[4.2.3] If the property change was a deletion, do nothing.

Example:

```
public class Example
{
    private static final String[] res =
    {
        "http://168.192.1.1/audio197.mp3",
        "http://10.0.0.1/contentdir?id=10"
    };

    private static final String[] resProtocolInfo =
    {
        "http-get:*:audio/mpeg:*",
        "http:*:audio/mp3:*"
    };

    private static final String[] resSize =
    {
        "3558000",
        null
    };

    public static void main(String[] args)
    {
        MetadataNode mn = MetadataNode.createMetadataNode();

        mn.addMetadata("didl-lite:res", res);
        mn.addMetadata("didl-lite:res@protocolInfo", resProtocolInfo);
        mn.addMetadata("didl-lite:res@size", resSize);
        mn.addMetadata("upnp:channelID", "");
        mn.addMetadata("upnp:channelID@type", "SI");

        /*
            XML equivalent:

            <res protocolInfo="http-get:*:audio/mpeg:*" size="3558000">
                http://168.192.1.1/audio197.mp3
            </res>
            <res protocolInfo="http:*:audio/mp3:*">
                http://10.0.0.1/contentdir?id=10
            </res>
            <upnp:channelID type="SI"/>
        */
    }
}
```

6.3.9.1.1 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" or one of the standardized namespaces, 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 section 2.4 of [XML] and section 7.2.5.9 of [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.

6.3.9.1.2 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>
```

6.3.9.2 getMetadata method

The format of the key argument is expected to be in the format defined in [UPNP CDS] section 2.2.20, with the exception that "::" is replaced by "#". For example, when passed the value "dc:title", the method will return the title in the Dublin Core namespace if the property is present.

For each property defined by UPNP, DLNA, [OCAP-HN], or this specification, the HNIMP SHALL return the property values from the getMetadata method as a java.lang.String object if the property is defined as single-valued and a java.lang.String[] object if the property is defined as multi-valued, unless specified otherwise by this specification or property constants that begin with "PROP_" defined in [OCAP-HN] Annex F.

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

For each vendor-defined property 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.

6.3.9.3 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.

6.3.9.4 createMetadataNode method

The HNIMP SHALL NOT use the key argument to the one-parameter form of this method.

6.3.9.5 getExtendedFileAccessPermissions method

The format of the key argument is expected to be in the format defined in [UPNP CDS] section 2.2.20, with the exception that "::" is replaced by "#".

6.3.9.6 getKey Method

When this method is called, the HNIMP SHALL return the value of the key parameter that can be utilized to retrieve this MetadataNode from this node's parent. The HNIMP SHALL include the namespace prefix in the value returned. DIDL-Lite is the default namespace in DLNA, so no prefix is required for DIDL-Lite defined properties in DLNA. Thus, for MetadataNode created from DLNA defined DIDL-Lite properties without a prefix, the HNIMP SHALL include "didl-lite" namespace prefix in the value returned.

6.3.9.7 getKeys Method

When this method is called, the HNIMP SHALL return values of key parameters for all metadata contained within this MetadataNode. The HNIMP SHALL include the namespace prefix in the values returned. For MetadataNode created from DLNA defined DIDL-Lite properties without a prefix, the HNIMP SHALL include "didl-lite" namespace prefix in values returned.

NOTE: Based on the requirements in Section 6.3.9, the HNIMP is required to explicitly specify namespace prefix when returning values for key parameters in getKey and getKeys methods.

6.3.10 VideoResource Interface

6.3.10.1 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.

To obtain the color depth value using org.ocap.hn.content.ContentResource.getResourceProperty(String key), the key parameter value SHALL be "res@colorDepth". ContentResource.getResourceProperty(String key) SHALL return a java.lang.Integer wrapping the same unsigned integer value returned by the VideoResource.getColorDepth() method. If the color depth is unknown, then a java.lang.Integer representing a value of -1 SHALL be returned.

6.3.10.2 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).

To obtain the resolution value using org.ocap.hn.content.ContentResource.getResourceProperty(String key), the key parameter value SHALL be “res@resolution”. ContentResource.getResourceProperty(String key) SHALL return a java.awt.Dimension matching the value returned by the VideoResource.getResolution() method. If the resolution is unknown, then null SHALL be returned.

6.3.11 StreamingActivityListener interface

6.3.11.1 notifyStreamingStarted method

When content streaming begins and the ContentItem corresponding to the content binary being streamed is of a type the application registered for when it called the ContentServerNetModule.addStreamingActivityListener method, an OC-DMS HNIMP SHALL call this method. Content streaming start is identified by an OC-DMS HNIMP when it is preparing to send or has sent the first transmission containing some or all of the requested content binary for an activity. An OC-DMS HNIMP MAY call this method in preparation of sending the transmission or concurrently with transmission commencement.

6.3.11.2 notifyStreamingChanged method

When the notifyStreamingStarted method was called for an activity and content streaming changes for that activity, the OC-DMS HNIMP SHALL call this method. Content streaming change is identified by the OC-DMS HNIMP when a different content binary URI is requested without ending the activity, e.g., play list streaming, or the CDS:X_DLNA_SelectChannel action is used to request a different channel ID from a virtualTuner.

6.3.11.3 notifyStreamingEnded method

When the notifyStreamingStarted method was called for an activity and content streaming ends for that activity, the OC-DMS HNIMP SHALL call this method. Content streaming end is identified by the OC-DMS HNIMP when the receiving device terminates the activity successfully using a transport protocol message, e.g., RTSP TEARDOWN, UPnP action, e.g., CM:ConnectionComplete, or times out when the server is waiting for a request for additional content. The timeout is OC-DMS HNIMP specific and is typically based on the transport protocol. Content streaming end is also identified by the OC-DMS HNIMP when a failure occurs that prevents further transmission of the content binary. For example, the consumer loses entitlement to view the content while streaming is in progress. The success and failure types are defined in the Java source for the StreamingActivityListener interface.

6.3.12 StreamableContentResource interface

6.3.12.1 getBitRate method

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

To obtain the bit rate using org.ocap.hn.content.ContentResource.getResourceProperty(String key), the key parameter value SHALL be “res@bitrate”. ContentResource.getResourceProperty(String key) SHALL return a java.lang.Long wrapping the same long value returned by the StreamableContentResource.getBitRate() method. If the bit rate is unknown, then a java.lang.Long representing a value of -1 SHALL be returned.

6.3.12.2 getDuration method

The getDuration method maps to the [DIDL-L] property res@duration, which is referenced in [UPNP CDS]. The HNIMP SHALL NOT set the res@duration property for locally hosted ContentEntrys that are not RecordingContentItems unless directed to do so by an application.

The HNIMP SHALL ensure that the res@duration property for a locally hosted RecordingContentItem that represents a recording in progress is set at the conclusion of the recording.

To obtain the duration using org.ocap.hn.content.ContentResource.getResourceProperty(String key), the key parameter value SHALL be “res@duration”. ContentResource.getResourceProperty(String key) SHALL return the

same javax.media.Time returned by the StreamableContentResource.getDuration() method. If the duration is unknown, null SHALL be returned.

6.3.13 ContentFormat Interface

This interface represents content metadata in the same format as fields in the <res> element protocolInfo parameter.

6.3.13.1 getContentProfile method

This method returns a DLNA media format profile identifier, or a vendor-specific media format profile identifier. When the ContentFormat is associated with a CDS content item <res> element with the DLNA.ORG_PN parameter set in the protocolInfo property, this method SHALL return the string value of that parameter, e.g., "AVC_TS_MP_HD_AC3_ISO", "DASH_MPD"

6.3.13.2 getProtectionType method

This method returns the content protection portion of DLNA profile identifier when content protection is being used. When the ContentFormat is associated with a CDS content item <res> element with the DLNA.ORG_PN parameter set in the protocolInfo property and the parameter includes a content protection prefix indicating link protection support, e.g., "DTCP", this method returns that prefix.

6.3.14 OutputVideoContentFormat Interface

This interface provides additional parameters for transforming video content.

6.3.14.1 getVerticalResolution method

This method communicates the maximum vertical resolution of the transformed content.

6.3.14.2 getHorizontalResolution method

This method communicates the maximum horizontal resolution of the transformed content.

6.3.14.3 getBitRate method

This method communicates the target bitrate of the transformed content.

6.3.14.4 isProgressive method

This method indicates if the transformed content will be interlaced or progressive.

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, indented 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, indented 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 NetSecurityManager.setAuthorizationHandler method, or if it was registered with the three-argument setAuthorizationHandler method and the notifyTransportRequests parameter was set to true, then the OC-DMS HNIMP SHALL call the notifyActivityStart method for every HTTP and RTSP message received for an activity that is accessing MSO content from the OC-DMS HNIMP. MSO content identification is defined in [HN-SEC].

When the NetAuthorizationHandler was registered using the three-argument setAuthorizationHandler method and the notifyTransportRequests parameter was set to false, then the OC-DMS HNIMP SHALL call the notifyActivityStart method for the first HTTP or RTSP message for an activity that is accessing MSO content from the OC-DMS HNIMP. Identification of the first message in an activity is defined in [HN-SEC].

If the message protocol is HTTP and an scid.dlna.org header is present in the request, the OC-DMS HNIMP SHALL set the activityID parameter to the ConnectionID value of the scid.dlna.org header. If the message protocol is RTP and an RTSP session-id header is present in the RTSP SETUP request, the OC-DMS HNIMP SHALL set the activityID parameter to the ConnectionID value of the session-id header. If no ConnectionID value is present in the HTTP or RTSP request, then the OC-DMS HNIMP SHALL generate a ConnectionID per the requirements of BCM (see Section 5.5) and set the activityID parameter to the ConnectionID value; if the NetAuthorizationHandler permits the operation, the OC-DMS HNIMP SHALL report the generated ConnectionID value in the HTTP or RTSP response.

When the notifyActivityStart method is called, the OC-DMS HNIMP SHALL NOT process the request until this method returns. If the request is denied, the OC-DMS HNIMP SHALL fail the request and return an error code 401 Not Authorized. If the request is granted, the OC-DMS 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 OC-DMS HNIMP SHALL call the notifyActivityEnd method when the activity terminates. Identification of activity termination is defined in [HN-SEC].

The OC-DMS HNIMP SHALL match activity end occurrences with activity start occurrences and pass the activity ID parameter to the notifyActivityEnd method that was previously provided in 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 OC-DMS HNIMP SHALL call the notifyAction method of that object whenever an action matching one of those names is received. When the action is CM:PrepareForConnection, the OC-DMS HNIMP SHALL set the activityID parameter to the value to be placed in the action ConnectionID OUT argument. When the action is CM:CompleteConnection or CM:GetCurrentConnectionInfo, the OC-DMS HNIMP SHALL set the activityID parameter to the value of the action ConnectionID IN argument. For all other actions, the OC-DMS HNIMP SHALL set the activityID parameter to the value of -1. The OC-DMS HNIMP SHALL NOT process the action until this method returns action accepted or denied. If this method returns denied, the OC-DMS HNIMP SHALL fail the action and return UPnP error code 606 Action Not Authorized.

6.6.2 NetAuthorizationHandler2 interface

6.6.2.1 notifyActivityStart method

When the NetAuthorizationHandler2 was registered using the overloaded NetSecurityManager.setAuthorizationHandler method that takes a NetAuthorizationHandler2 parameter and the notifyTransportRequests parameter was set to true, then the OC-DMS HNIMP SHALL call the notifyActivityStart method for every HTTP and RTSP message received for an activity that is accessing MSO content from the OC-DMS HNIMP. MSO content identification is defined in [HN-SEC].

When the notifyTransportRequests parameter was set to false, then the OC-DMS HNIMP SHALL call the notifyActivityStart method for the first HTTP or RTSP message for an activity that is accessing MSO content from the OC-DMS HNIMP. Identification of the first HTTP or RTSP message in an activity is defined in [HN-SEC].

If the message protocol is HTTP and an scid.dlna.org header is present in the request, the OC-DMS HNIMP SHALL set the activityID parameter to the ConnectionID value of the scid.dlna.org header. If the message protocol is RTP and an RTSP session-id header is present in the RTSP SETUP request, the OC-DMS HNIMP SHALL set the activityID parameter to the ConnectionID value of the session-id header. If no ConnectionID value is present in the HTTP or RTSP request, then the OC-DMS HNIMP SHALL generate a ConnectionID per the requirements of BCM (see Section 5.5) and set the activityID parameter to the ConnectionID value; if the NetAuthorizationHandler permits the operation, the OC-DMS HNIMP SHALL report the generated ConnectionID value in the HTTP or RTSP response.

When the notifyActivityStart method is called, the OC-DMS HNIMP SHALL determine if the url parameter uniquely matches the URI value of a content item res property in the OC-DMS HNIMP CDS, and if it does, the OC-DMS HNIMP SHALL set the entry parameter to an object reference corresponding to the content item. If no content item is found, the OC-DMS HNIMP SHALL set the value null.

The OC-DMS HNIMP SHALL set the request parameter to the message request. In that case, when the protocol is HTTP, the first String in the array SHALL be set to the HTTP Request-Line, and subsequent Strings in the array SHALL be set to the HTTP headers in the order they were received. Trailing CRLF characters, the blank line following the HTTP headers, and the HTTP message body SHALL NOT be included in the array. (See [RFC 2616].) When the protocol is RTSP, the OC-DMS HNIMP SHALL set the request parameter to an empty array.

When the notifyActivityStart method is called in response to a network request, the OC-DMS HNIMP SHALL NOT process the request until this method returns. If the notifyActivityStart method returns false, the OC-DMS HNIMP SHALL fail the request and return an error code 401 Not Authorized. If the notifyActivityStart method returns true, the OC-DMS HNIMP SHALL stream the requested content to the client.

6.6.2.2 notifyActivityEnd method

When an application has set a NetAuthorizationHandler2 object by calling the overloaded NetSecurityManager.setAuthorizationHandler method that takes a NetAuthorizationHandler2 parameter, the OC-DMS HNIMP SHALL call the notifyActivityEnd method when the activity terminates. Identification of activity termination is defined in [HN-SEC].

The OC-DMS HNIMP SHALL match activity end occurrences with activity start occurrences and pass the activityID parameter to the notifyActivityEnd method that was previously provided in the corresponding call to the notifyActivityStart method.

The OC-DMS HNIMP SHALL set the resultCode parameter as follows:

1. If the activity was terminated by a message from the OC-DMS HNIMP to a remote device, then set the resultCode to the response value returned in the message. This will be one of the HTTP, RTSP, or UPnP response codes.
2. If the activity was terminated by a detected network failure or timeout, set the resultCode to 408 Request Timeout.

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.

6.6.2.3 notifyAction method

When an application has set a NetAuthorizationHandler2 object by calling the overloaded NetSecurityManager.setAuthorizationHandler method that takes a NetAuthorizationHandler2 parameter and passes in one or more action names to that method, then the OC-DMS HNIMP SHALL call the notifyAction method of that object whenever an action matching one of those names is received.

When the action is CM:PrepareForConnection, the OC-DMS HNIMP SHALL set the activityID parameter to the value to be placed in the action ConnectionID OUT argument. When the action is CM:CompleteConnection or CM:GetCurrentConnectionInfo, the OC-DMS HNIMP SHALL set the activityID parameter to the value of the action ConnectionID IN argument. For all other actions, the OC-DMS HNIMP SHALL set the activityID parameter to the value of -1.

The OC-DMS HNIMP SHALL set the request parameter to the message request. In that case, when the protocol is HTTP, the first String in the array SHALL be set to the HTTP Request-Line, and subsequent Strings in the array SHALL be set to the HTTP headers in the order they were received. Trailing CRLF characters, the blank line following the HTTP headers, and the HTTP message body SHALL NOT be included in the array. (See [RFC 2616].) When the protocol is RTSP, the OC-DMS HNIMP SHALL set the request parameter to an empty array.

The OC-DMS HNIMP SHALL NOT process the action until this method returns true or false. If this method returns false, the OC-DMS HNIMP SHALL fail the action and return UPnP error code 606 Action Not Authorized.

6.6.3 NetSecurityManager class

6.6.3.1 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 OC-DMS HNIMP SHALL stop the stream and send messages to the recipient appropriate for the transport protocol being used.

6.6.3.2 setAuthorizationHandler overloaded method

For the overloaded setAuthorizationHandler method with the actionNames parameter, the parameter is either empty array or 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 OC-DMS 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.

6.6.3.3 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 6.6.3.2.

6.7 Package org.ocap.hn.service

6.7.1 RemoteService interface

No UPnP mapping is defined for RemoteService interface.

6.7.2 ServiceResolutionHandler interface

6.7.2.1 notifyTuneFailed method

The OC-DMS HNIMP SHALL call this method when a tuning request to stream a ChannelContentItem over the network fails for any reason.

6.7.2.2 resolveChannelItem method

The OC-DMS HNIMP SHALL call this method when a request to stream a ChannelContentItem over the network is received but the ChannelContentItem does not contain tuning parameters.

6.7.3 MediaServerManager class

6.7.3.1 getHttpMediaPortNumber() method

The getHttpMediaPortNumber method returns the port number assigned to the <res> value (URI) in all audio and video content items that are streamed over HTTP.

6.7.4 HttpRequestResolutionHandler interface

6.7.4.1 resolveHttpRequest method

The HNIMP OC-DMS calls this method to resolve HTTP GET or HEAD requests to a <res> URI as per Section 5.6.2.

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 6.8.1.2 deleteEntry method

When the deleteEntry method is invoked, the HNIMP SHALL perform the same steps as identified in the mapping of ContentContainer.removeContentEntry method with ContentEntry input parameter of NetRecordingEntry.

6.8.1.3 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.

6.8.1.4 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.

6.8.1.5 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 and remote modification of that recording is not disallowed by requirements in Section C.1.1.8, 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 6–3 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 6–3. 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 6–3. 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 NetRecording Entry. 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 OcapRecordingRequest state changes, the implementation SHALL map the OcapRecordingRequest state to the RecordTask state as shown in Table 6–2. Changes in state of the OcapRecordingRequest map to changes in the ocap:taskState property of the RecordingContentItem. Whenever CDS properties in the OCAP name space change, the HNIMP SHALL generate a CDS changed event.

Table 6–2 - Recording Request State Mapping

OCAP RECORDING REQUEST STATE VALUE	OCAP RECORDING REQUEST STATE NAME	UPNP SRS RECORD TASK STATE
1	PENDING_NO_CONFLICT_STATE	IDLE.READY
2	PENDING_WITH_CONFLICT_STATE	IDLE.ATRISK
4	IN_PROGRESS_STATE	ACTIVE.TRANSITION.FROMSTART
16	IN_PROGRESS_INCOMPLETE_STATE	ACTIVE.TRANSITION.RESTART
15	IN_PROGRESS_WITH_ERROR_STATE	ACTIVE.NOTRECORDING
4	IN_PROGRESS_STATE	ACTIVE.RECORD.FROMSTART.OK
5	IN_PROGRESS_INSUFFICIENT_SPACE_STATE	ACTIVE.RECORDING.FROMSTART.ATRISK
16	IN_PROGRESS_INCOMPLETE_STATE	ACTIVE.RECORDING.RESTART.OK
<NO MAPPING>	<NO MAPPING>	ACTIVE.RECORDING.RESTART.ATRISK
7	COMPLETED_STATE	DONE.FULL
6	INCOMPLETE_STATE	DONE.PARTIAL
8	FAILED_STATE	DONE.EMPTY
10	CANCELLED_STATE	DONE.EMPTY

OCAP RECORDING REQUEST STATE VALUE	OCAP RECORDING REQUEST STATE NAME	UPNP SRS RECORD TASK STATE
9	TEST_STATE	
14	DELETED_STATE	DONE.EMPTY

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 6.3.9.1.2. 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 ocap:scheduledChannelIDType 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 6.3.9.1.2 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 6–3 - 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. The OC-DMS HNIMP SHALL populate the res property from the required ocap:contentURI. If a RecordingRequest is deleted, HNIMP SHALL delete the corresponding RecordingContentItem, any associated metadata, and any SRS and CDS entries.

If the RecordedService(s) associated with a RecordingRequest are deleted, the HNIMP SHALL delete the ContentResource objects and resource metadata from the corresponding RecordingContentItem.

NOTE: Calling RecordingContentItem.deleteEntry() only deletes the corresponding RecordingContentItem, any associated metadata, and any SRS and CDS entries - not the corresponding recording on the DVR server. Therefore, in order to properly delete a recording from a DVR server, applications must invoke RecordingRequest.delete() rather than RecordingContentItem.deleteEntry().

6.8.5.1 deleteEntry method

When the deleteEntry method is invoked, the HNIMP SHALL perform the same steps as identified in the mapping of ContentContainer.removeContentEntry with the ContentEntry input parameter of RecordingContentItem.

6.8.5.2 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.

6.8.5.3 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, provided remote modification of that RecordingContentItem is not disallowed by requirements in Section C.1.1.8.

6.8.5.4 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.

6.8.5.5 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 ClientNegotiations

   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)

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

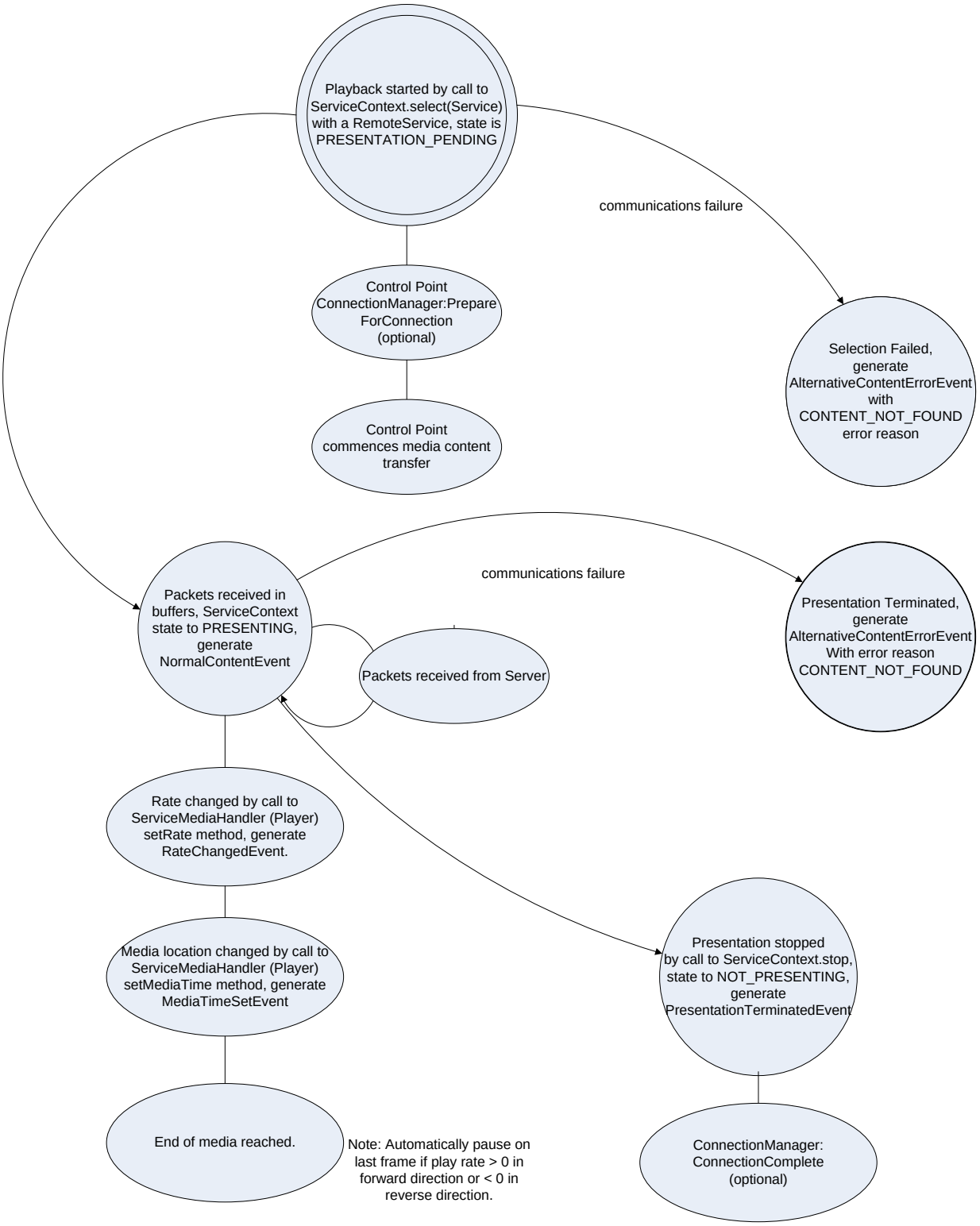

Figure 6–1 - ServiceContext States during RemoteService Presentation

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) for all content items that support a DLNA Random Access Data Availability Model, as defined in Section 5.2. 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.

6.11 Package org.ocap.hn.ruihsrc

6.11.1 RemoteUIServerManager class

6.11.1.1 setUIList method

The setUIList method sets the remote user interface description XML document used in the UPnP Remote UI Server Service defined in [UPNP RUISS]. Until this method is called, an HNIMP OC-DMS SHALL return an empty list in a response to the GetCompatibleUIs action; see section 2.4.1 of [UPNP RUISS]. An HNIMP OC-DMS SHALL NOT persist a remote user interface description set by this method across reboot or power-cycle.

When an application calls the setUIList method, the HNIMP OC-DMS SHALL,

1. throw an IllegalArgumentException if the XML document is not null and is detected to be invalid as per the schema defined in [UPNP RUISS] section 4.
2. remove any existing uiList document that was previously set.
3. set the uiList XML document as the new source against which matching remote user interfaces are sent in response to a GetCompatibleUIs action; [UPNP RUISS] section 2.4.1. If the uiList parameter is a null, an empty list will be sent in response to subsequent GetCompatibleUIs actions.
4. send the UIListingUpdate event; [UPNP RUISS] section 2.2.1.

6.12 Package org.ocap.hn.transformation

6.12.1 TransformationManager abstract class

This class can be used to cause a content binary to be transformed, and CDS content item <res> elements to be published representing the transformed content. The API allows content to be transformed to HTTP Adaptive streaming profiles defined in [DLNA HTTP-AD].

When an HNIMP OC-DMS publishes a CDS <res> element that represents HTTP Adaptive content, it SHALL populate the res@protocolInfo field as defined in [DLNA HTTP-AD].

6.12.1.1 setDefaultTransformations method

When the setDefaultTransformations method is invoked, the HNIMP OC-DMS SHALL set those transformations to be applied by default to all newly-created server local MSO content. Subsequent calls to the method SHALL replace the existing default transformations.

6.12.1.2 getDefaultTransformations method

Returns the transformations that have been configured by a call to the setDefaultTransformations method.

6.12.1.3 getSupportedTransformations method

When the getSupportedTransformations method is invoked, the HNIMP OC-DMS creates an array of Transformation objects that match the set of transformations the device is capable of representing with the ci-param flag in the 4th field of the protocolInfo property. The set of Transformation objects returned is implementation-specific.

6.12.1.4 getTransformations method

The getTransformations method provides the existing transformations that are applied to the content item.

6.12.1.5 setTransformations method

When the setTransformations method is invoked, the HNIMP OC-DMS SHALL add <res> elements that correspond to supported transformations to each applicable ContentItem object in the array parameter that represents server local MSO content. The <res> elements that correspond to a transformation are identified by the presence of the ci-param flag in the 4th argument of the protocolInfo property.

Invoking a setTransformations method that explicitly takes the Transformation array parameter removes any existing transformations before applying the new transformations.

If any affected ContentItem object is associated with a published CDS content item, then the HNIMP OC-DMS SHALL make the same changes to the CDS content item.

6.12.1.6 addTransformationListener method

This method allows an application to register a listener and receive callbacks from the HNIMP OC-DMS when transformations are being applied to ContentItems. An application can register multiple unique listeners using this method.

6.12.1.7 removeTransformationListener method

This method allows an application to remove a previously registered listener and stop receiving callbacks.

6.12.2 Transformation Interface

This interface represents the content transformation capability of the implementation or communicates a desired content transformation to the implementation.

6.12.2.1 getInputContentFormat method

This method returns the ContentFormat of the source content.

6.12.2.2 getOutputContentFormats method

This method returns an array of ContentFormat objects.

6.12.3 TransformationListener Interface

An application can implement this interface to receive callbacks from the TransformationManager as transformations are being applied to content items. An application SHALL receive one and only one notification for each requested ContentFormat transformation.

6.12.3.1 notifyTransformationReady method

The HNIMP OC-DMS SHALL call this method upon creation of each ContentResource representing the output ContentFormat of a transformation being applied to a ContentItem. Upon receiving this notification, the application can assume that the <res> elements representing the transformation associated with the content item have been added.

6.12.3.2 notifyTransformationFailed method

The HNIMP OC-DMS SHALL call this method with an appropriate reason code if a transformation being applied to a ContentItem fails.

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] 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>
      <modelNumber>model number</modelNumber>
      <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_DLNADOC xmlns:dlna="urn:schemas-dlna-org:device-1-0">
         DMS-1.50
      </dlna:X_DLNADOC>
      <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:device-1-
0">
         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>
         <service>
            <serviceType>urn:cablelabs-
com:service:ViewPrimaryOutputPort:1</serviceType>
            <serviceId>urn:cablelabs-
```

```
com:serviceId:ViewPrimaryOutputPort</serviceId>
            <SCPDURL>ViewPrimaryOutputPort.xml</SCPDURL>
            <eventSubURL>ViewPrimaryOutputPort.xml/Event</eventSubURL>
            <controlURL>ViewPrimaryOutputPort.xml/Control</controlURL>
         </service>
      </serviceList>
   </device>
</root>
```

A.3 ViewPrimaryOutputPort Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>
   <actionList>
      <action>
         <name>Tune</name>
         <argumentList>
            <argument>
               <name>ConnectionID</name>
               <direction>in</direction>

   <relatedStateVariable>A_ARG_TYPE_ConnectionID</relatedStateVariable>
            </argument>
            <argument>
               <name>TuneParameters</name>
               <direction>in</direction>

   <relatedStateVariable>X_ARG_TYPE_TuneParameters</relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>AudioMute</name>
         <argumentList>
            <argument>
               <name>ConnectionID</name>
               <direction>in</direction>

   <relatedStateVariable>A_ARG_TYPE_ConnectionID</relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>AudioRestore</name>
         <argumentList>
            <argument>
               <name>ConnectionID</name>
               <direction>in</direction>

   <relatedStateVariable>A_ARG_TYPE_ConnectionID</relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>PowerOn</name>
      </action>
      <action>
```

```
         <name>PowerOff</name>
         <argumentList>
            <argument>
               <name>ConnectionID</name>
               <direction>in</direction>

   <relatedStateVariable>A_ARG_TYPE_ConnectionID</relatedStateVariable>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>PowerStatus</name>
         <argumentList>
            <argument>
               <name>PowerStatus</name>
               <direction>out</direction>

   <relatedStateVariable>X_ARG_TYPE_PowerStatus</relatedStateVariable>
            </argument>
         </argumentList>
      </action>
   </actionList>
   <serviceStateTable>
      <stateVariable sendEvents="no">
         <name>X_ARG_TYPE_TuneParameters</name>
         <dataType>string</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>A_ARG_TYPE_ConnectionID</name>
         <dataType>i4</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>X_ARG_TYPE_PowerStatus</name>
         <dataType>string</dataType>
         <allowedValue>FULL POWER</allowedValue>
         <allowedValue>STANDBY</allowedValue>
      </stateVariable>
   </serviceStateTable>
</scpd>
```

Annex B Use of Trick Modes Headers

When making a trick mode request, an HNIMP OC-DMP may explicitly request the server to re-stamp the PTS and PCR. A client may also implicitly assume that the PTS and PCR in the response stream are re-stamped based on the absence of the FrameRateInTrickMode.dlna.org header. Subsequently when responding to a trick mode request, an HNIMP OC-DMS may not re-stamp the PTS and PCR and instead indicate to the client the frame rate at which the response is to be played out. The server will use either the FrameRateInTrickMode.ochn.org or the FrameRateInTrickMode.dlna.org header depending on the client request. The following diagram clarifies the conditions under which the HNIMP OC-DMS includes the various headers in the trick mode response.

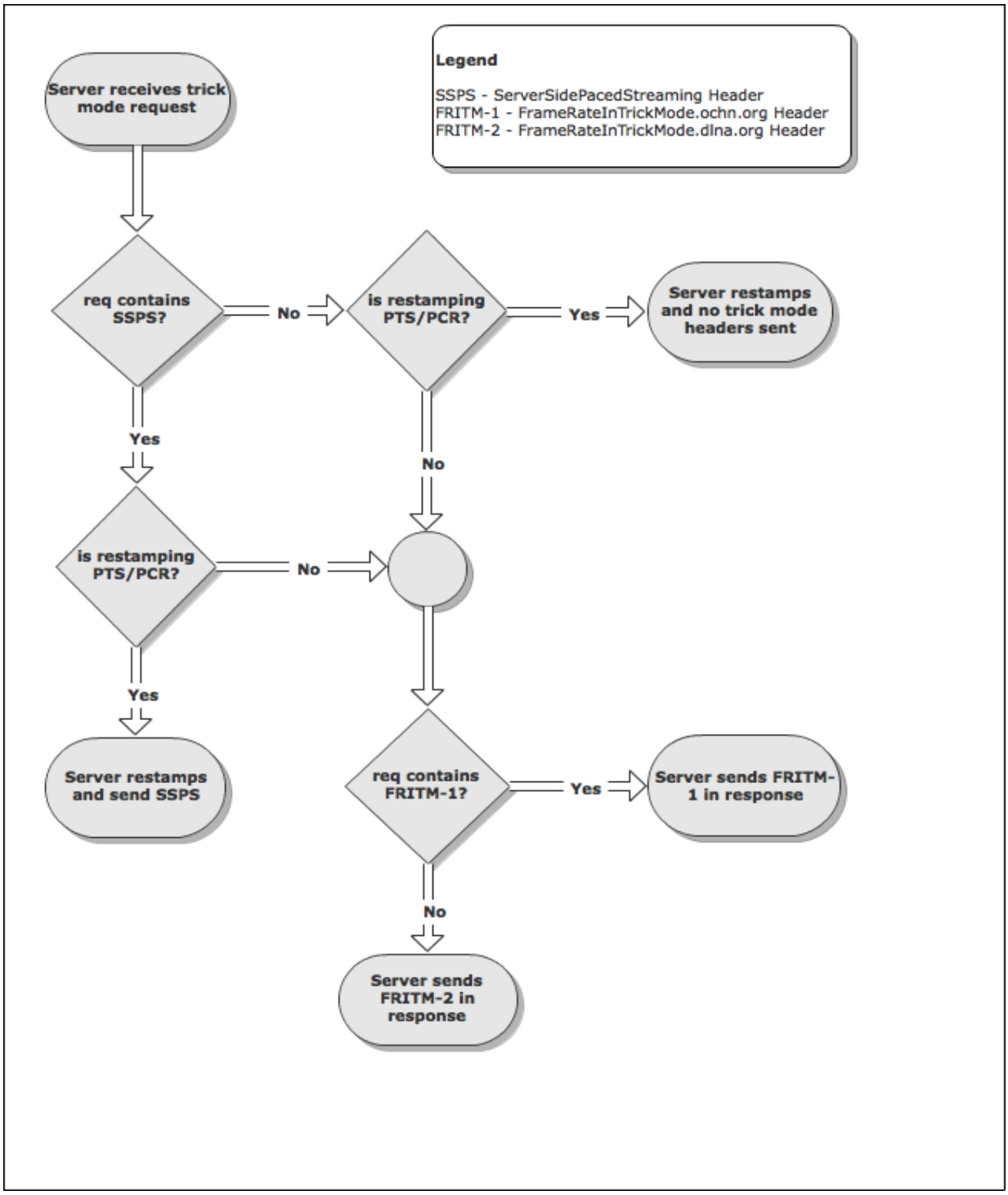

Figure B–1 - Trick Mode Response

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 6–3 are also used for requesting the scheduled recording.

In order to ensure interoperability with older implementations, the HNIMP SHALL provide both ocap:scheduleStartDateTime and ocap:scheduledStartDateTime property keys in both the RecordingContentItem and CDS videoItem when either is present. The HNIMP SHALL guarantee identical values for both ocap:scheduleStartDateTime and ocap:scheduledStartDateTime property keys when either one is present. It SHALL be implementation-dependent as to which value the HNIMP uses if both ocap:scheduleStartDateTime and ocap:scheduledStartDateTime property keys are set with differing values.

NOTE: This mapping does not extend to any SRS mapping requirement, which remains ocap:scheduledStartDateTime to srs:scheduledStartDateTime.

C.1.1.1 Task State

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

Description: The taskState property indicates the current state of the recording for both the SRS RecordTask and the OCAP RecordingRequest. The value SHALL be one of the integer state constants of the org.ocap.dvr.OcapRecordingRequest interface.

Default Value: N/A – Output Only

Sort Order: Sequenced Numeric

Property Key: ocap:taskState

C.1.1.2 Start Time

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

Description: The start time 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 and ocap:scheduleStartDateTime

C.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]. This property is converted to an Integer when accessed from the MetadataNode.getMetadata(String) method with a parameter of PROP_DURATION.

Default Value: N/A – Output only.

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

Property Key: ocap:scheduledDuration

C.1.1.4 Source (OCAP Locator)

Namespace: ocap **Property Data Type:** xsd:string **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.1.1.5 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.1.1.6 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.1.1.7 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.1.1.8 Access Permissions

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

Description: The accessPermissions property indicates the permissions for reading or writing the content. This specification distinguishes between remote access (access via a CDS action) and local access (all other access).

For remote access:

1. The HNIMP SHALL only honor the world-read and world-write permission flags and SHALL ignore all other permission flags.
2. Content metadata and content binary may be modified remotely only if world-write access is granted. An attempt to modify content metadata or content binary remotely when world-write access is denied SHALL result in UPnP error 501 (Action Failed).
3. The HNIMP SHALL NOT allow remote modifications to accessPermissions property.
4. If accessPermissions property and the @restricted property are both present and the @restricted property is set to 1, then the @restricted property SHALL override accessPermissions property for purposes of write access.

For local access:

1. The HNIMP SHALL honor all permission flags.
2. If the ocap:appID property is present, then the HNIMP SHALL use it to determine whether to grant or deny owner-read, owner-write, organization-read, and organization-write access; otherwise, the HNIMP SHALL deny such access.
3. The accessPermissions property may be modified after it is added to the CDS database; the world-write and world-read flags may be changed. In this case, the CDS eventing SHALL reflect the remote view of the CDS database, e.g., if world-read flag for an item is changed from ‘0’ to ‘1’, the CDS eventing SHALL reflect ‘item added;' event.

For both remote and local access:

1. If the world-read flag is set to '0' for a content item or container, then:
    a. the implementation SHALL NOT allow the item or container to be discovered or read by non-privileged applications (i.e., those without owner_app, owner_org or other_org privileges) via any CDS action, (e.g., CDS:Search).
    b. the implementation SHALL NOT generate any events for the content item, e.g., CDS LastChange.
2. If the world-read is set to '0' for a content container, then world-read access SHALL NOT be granted to any content item or container contained within that container regardless of the world-read flag value of the

contained item or container. This overrides the access permission rules defined above and is necessary to ensure that content items or containers within a hidden container remain hidden.

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 and devices not covered by the owner_app, owner_org or other_org permissions.
owner_org	Read and write permission for applications in the same organization of the owner application. The value of this field (represented by 'hex_string' in the BNF) is no longer used but is retained for backward compatibility. The HNIMP SHALL populate it in accordance with the format specified above with an implementation-specific value. However, the HNIMP SHALL ignore its value for granting owner access.
owner_app	Read and write permissions for the application that owns the content. The value of this field (represented by 'hex_string' in the BNF) is no longer used but is retained for backward compatibility. The HNIMP SHALL populate it in accordance with the format specified above with an implementation-specific value. However, the HNIMP SHALL ignore its value for granting owner access.
other_org	Read and write permissions for organizations other than the owning organizations. This is an optional field, and any number of organizations MAY be listed. This value will match the organization_id in the application_identifier signaling found in an AIT or XAIT.
read	Read access is granted or denied depending on whether the value is "1" or "0", respectively (except as noted above).
write	Write access is granted or denied depending on whether the value is "1" or "0", respectively (except as noted above).

This property is converted to an org.ocap.storage.ExtendedFileAccessPermissions object when accessed from the metadataNode.getMetadata(String) method with a parameter of PROP_ACCESS_PERMISSIONS.

Default Value: N/A – Output Only

Sort Order: N/A multi-valued

Property Key: ocap:accessPermissions

C.1.1.9 Organization

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:organization

C.1.1.10 Application Identifier

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.1.1.11 Space Required

Namespace: ocap **Property Data Type:** xsd:long **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.1.1.12 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.1.1.13 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 6.8.5.3.

Default Value: N/A – Output Only

Sort Order: Sequenced Numeric

Property Key: ocap:mediaPresentationPoint

C.1.1.14 MSO Recorded Content Indicator

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.1.1.15 Expiration Period

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

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

Default Value: N/A – Output Only

Sort Order: Sequenced Numeric

Property Key: ocap:expirationPeriod

C.1.1.16 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.1.1.17 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:sheduledChannelID@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

Property Key: ocap:scheduledCDSEntryID

C.1.2.2 CDS Reference

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:scheduledStartDateTime>20090503T19:00:00</ocap:scheduledStartDateTime>
⁢ocap:scheduledDuration>P01:00:00</oca: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>
&lt/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 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.1.3.2 RecordingContentItem List

Namespace: ocap **Property Data Type:** 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.1.4 Definitions of OCAP-extended properties of CDS item class for Live Streaming

The following properties are used to indicate capabilities associated with Live Streaming.

C.1.4.1 Alternate URI

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

Description: An application MAY add this property to provide a URI that is an alternate to a <res> property URI value. This property is a dependent property of a <res> property. See [UPNP CDS] for dependent property definition. During a content access, e.g., HTTP GET, if the <res> URI value does not match the URI in the request and the <res> property contains a res@ocap:alternateURI property, then the implementation SHALL compare the path component of the alternateURI to the path component of the request URI (i.e., the scheme, authority, query, and fragment components are not used in the comparison). If the two path components are identical, the HNIMP SHALL respond as if the <res> property URI value matched the request. If the request URI or the alternate URI are not compliant with [RFC 3986], no comparison is performed.

When authorizing with the Network Authorization Handler, the implementation SHALL provide the original HTTP request URI (including the URI query component, if present) to the handler.

Default Value: N/A

Property Key: res@ocap:alternateURI

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–1 - 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–2 - 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

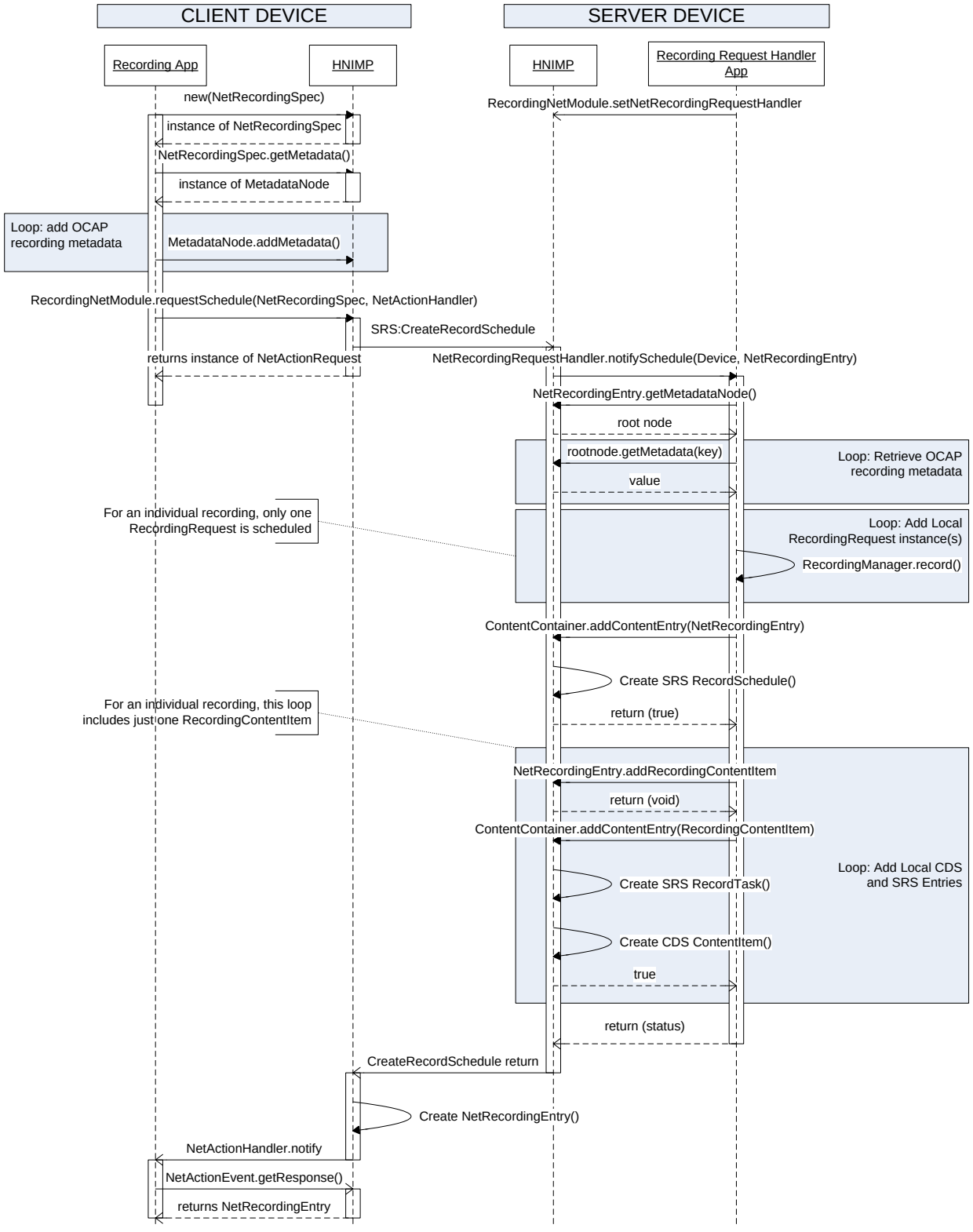

Figure I–1 - Recording Schedule Sequence Diagram 1

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 user 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.addContentEnty() 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

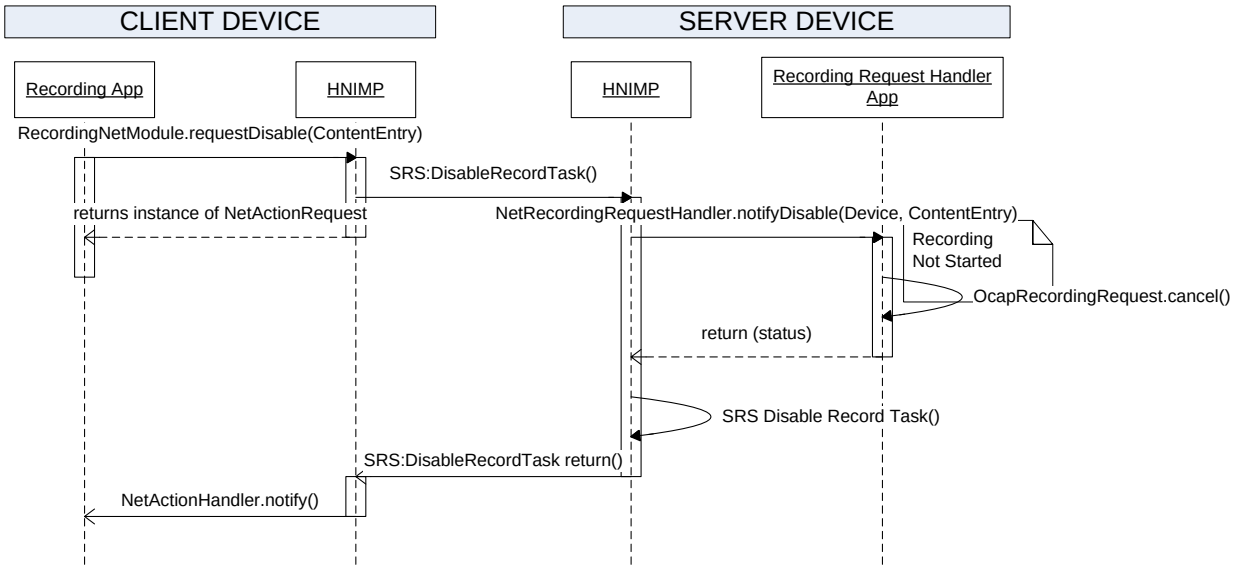

Figure I–2 - Cancel Individual Recording Sequence Diagram

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

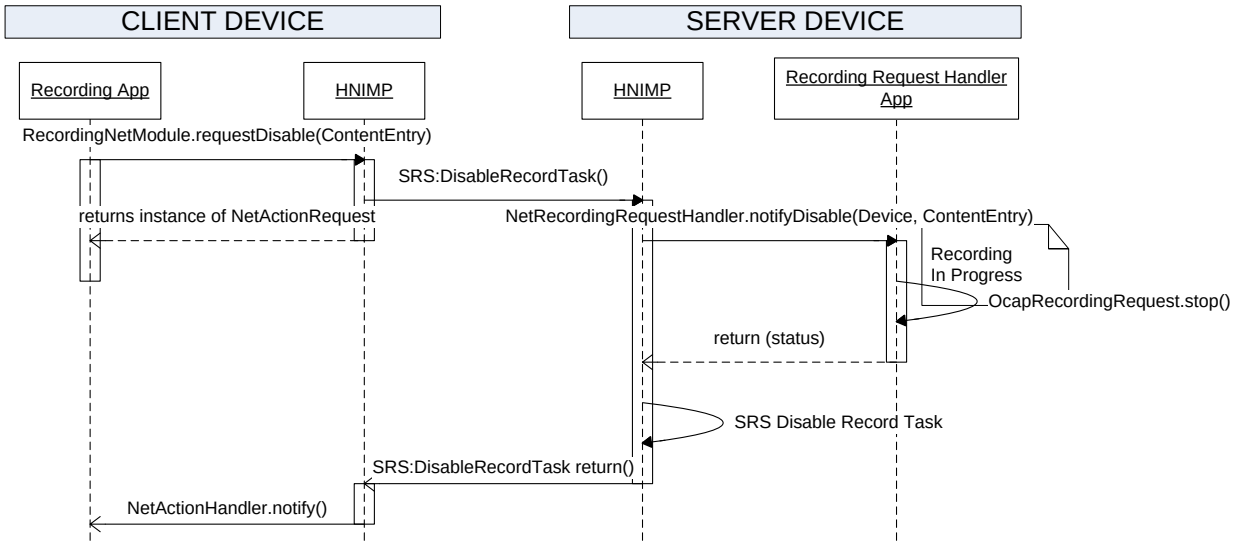

Figure I–3 - Stop Individual Recording Diagram

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

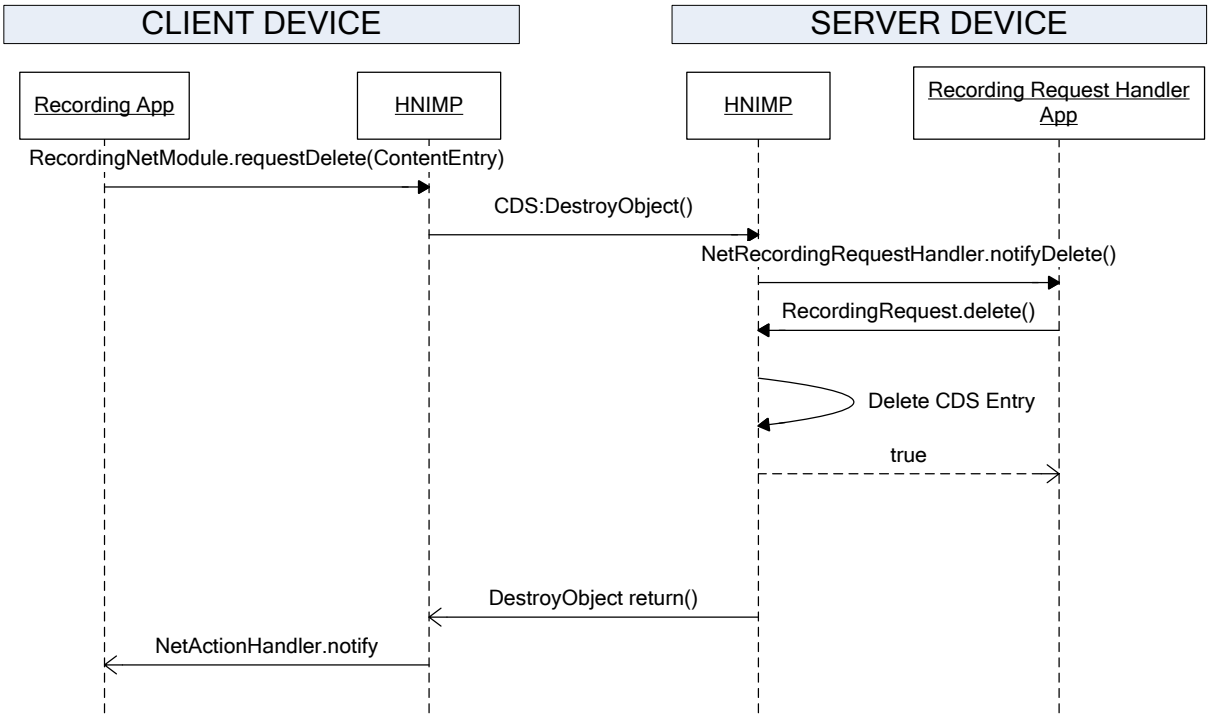

Figure I–4 - Delete Individual Recording Diagram

I.5 Schedule Local Recording

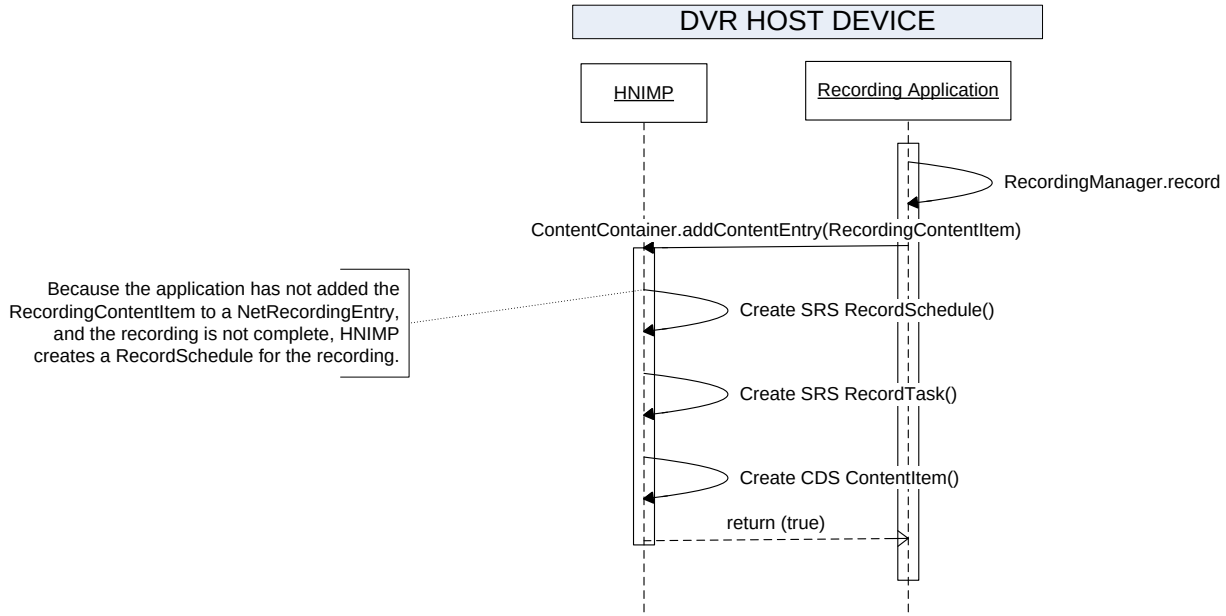

Figure I–5 - Schedule Local Recording Diagram

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 require 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

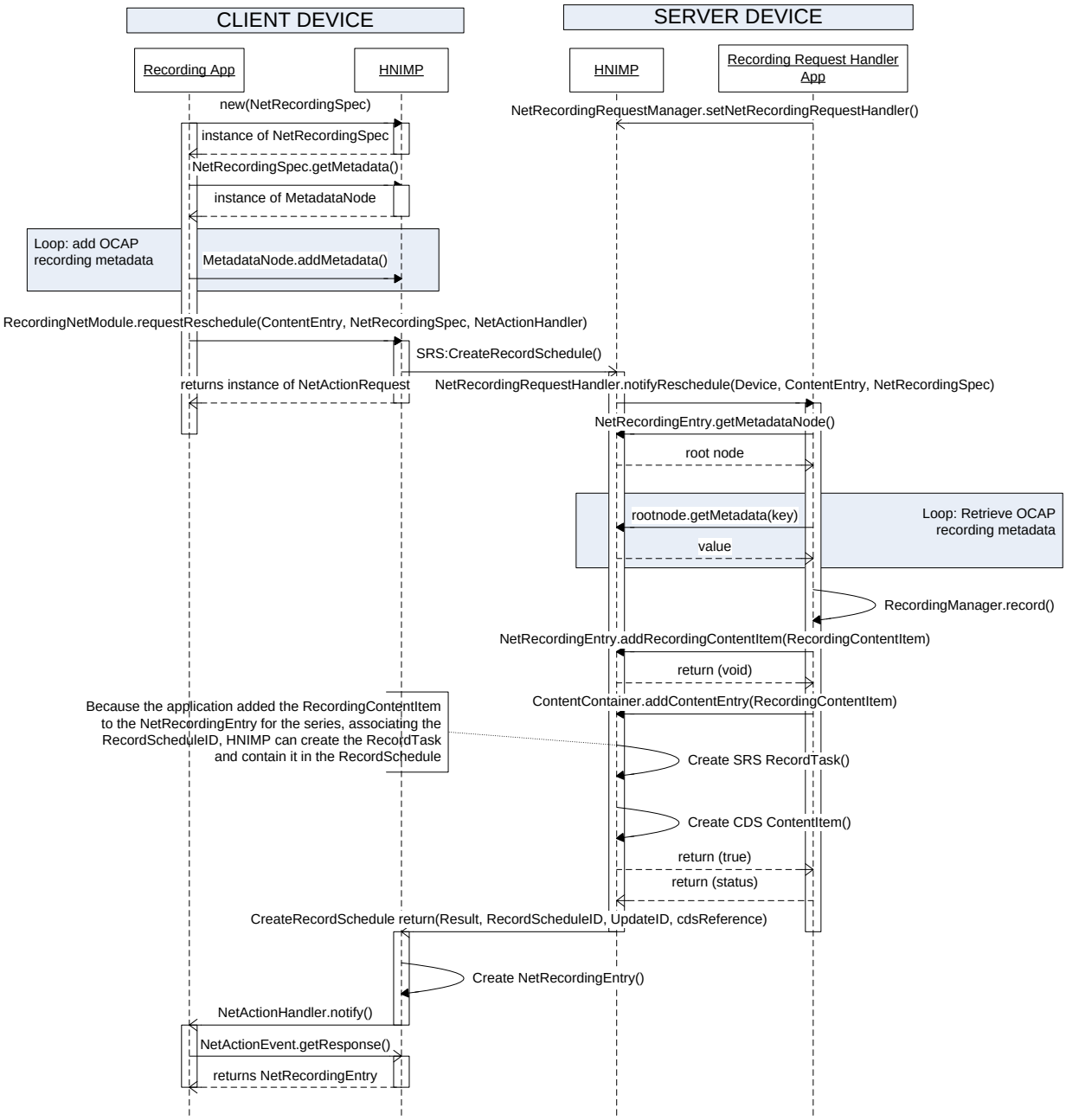

Figure I–6 - Add Recording to Series Diagram

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 user 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. ContentContainter.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

The following ECNs were incorporated into version I04 of this specification:

EC Identifier	Accepted Date	Title of EC
HNP2.0-N-08.1300-3	2/24/11	MaxBandwidth HTTP Header Removal
HNP2.0-N-10.1597-1	2/24/11	Clarification of XML encoding rules
HNP2.0-N-10.1598-1	2/24/11	ocap:schedule[d]StartDateTime interoperability
HNP2.0-N-10.1599-1	2/24/11	Clarification of allowed object types in RecordingContentItem properties
HNP2.0-N-10.1600-1	2/24/11	Clarification of type-SI channel ID format
HNP2.0-N-10.1601-1	2/24/11	Defining ocap:taskState as integer
HNP2.0-N-10.1605-1	2/24/11	HNP: Clarify type of spaceRequired
HNP2.0-N-10.1607-1	2/24/11	Removing support for Media Storage Volume
HNP2.0-N-10.1609-1	2/24/11	Clarification about HN Content Format support
HNP2.0-N-10.1614-3	2/24/11	Clarification for deleting recording content item
HNP2.0-N-10.1615-2	2/24/11	Publishing Model Clarification
HNP2.0-N-10.1632-1	2/24/11	Namespace Clarifications
HNP2.0-N-10.1633-1	2/24/11	Clarification of setting res@size and res@duration properties
HNP2.0-N-11.1636-2	2/24/11	EFAP Clarification
HNP2.0-N-11.1638-1	2/24/11	Resolution of inter-spec contradictions about metadata property types
HNP2.0-N-11.1640-2	2/24/11	Multi-valued Metadata

The following ECN was incorporated into version I05 of this specification:

EC Identifier	Accepted Date	Title of EC
HNP2.0-N-11.1659-4	5/12/11	HNP Reference edits for OpenCable bundle inclusion

The following ECNs were incorporated into version I06 of this specification:

EC Identifier	Accepted Date	Title of EC
HNP2.0-N-10.1613-2	1/12/12	HNP Copy Text
HNP2.0-N-11.1689-1	1/12/12	OC-DMS and OC-DMP compliance with DLNA QoS guidelines
HNP2.0-N-11.1694-2	1/12/12	HNP Live Streaming to DLNA 1.5 Devices
HNP2.0-N-11.1713-1	1/12/12	Hidden Container HNP
HNP2.0-N-11.1739-1	1/12/12	Alternate URI
HNP2.0-N-12.1747-1	1/12/12	HNP Service Resolution Handler

The following ECNs were incorporated into version I07 of this specification:

EC Identifier	Accepted Date	Title of EC
HNP2.0-N-12.1752-1	2/24/12	Parental Control Rating UPnP Property
HNP2.0-N-12.1760-3	2/24/12	View Primary Output Port

The following ECNs were incorporated into version I08 of this specification:

EC Identifier	Accepted Date	Title of EC
HNP2.0-N-11.1676-3	5/31/12	Update Media Format Requirements
HNP2.0-N-12.1783-2	5/31/12	URI Port Number Requirements
HNP2.0-N-12.1785-2	5/31/12	HTML5 RUI Server Compliance
HNP2.0-N-12.1789-1	5/31/12	Alternate URI Query
HNP2.0-N-12.1791-1	5/31/12	UPnP Rating Type Correction

The following ECNs were incorporated into version I09 of this specification:

EC Identifier	Author	Accepted Date	Title of EC
HNP2.0-N-12.1796-1	Ladd	4/18/2013	View Primary Output Port OrgID
HNP2.0-N-12.1802-1	Allanki	4/18/2013	Editorial changes to Media Format Requirements
HNP2.0-N-12.1803-1	Taylor	4/18/2013	Restamping in trick modes
HNP2.0-N-13.1816-1	Ladd	4/18/2013	HNP - Transformation API Mapping
HNP2.0-N-13.1817-2	Thakore	4/18/2013	HNP - Diagnostics EndPoint
HNP2.0-N-13.1826-1	Millard	4/18/2013	Update TuneParameters in VPOP SCPD
HNP2.0-N-13.1831-1	Millard	4/18/2013	VPOP Streaming Request Clarification

The following ECNs were incorporated into version I10 of this specification:

EC Identifier	Author	Accepted Date	Title of EC
HNP2.0-N-13.1818-1	Millard	5/30/13	Remove RecordingContentItem from NetRecordingEntry
HNP2.0-N-13.1829-1	Millard	5/30/13	NetManager NetList Query
HNP2.0-N-13.1840-1	Millard	5/30/13	ContentResource getResourceProperty Clarifications
HNP2.0-N-13.1841-1	Yelekyathanhalli	5/30/13	Hidden Container Update
HNP2.0-N-13.1842-1	Millard	5/30/13	Removing ContentResources When RecordedService is Deleted
HNP2.0-N-13.1843-1	Millard	5/30/13	Incomplete URN for OCAP Namespace
HNP2.0-N-13.1846-1	Ladd	5/30/13	Preferred IP Address HNP
HNP2.0-N-13.1851-1	Pratt	5/30/13	New ChannelContentItem type VIDEO_ITEM_BROADCAST_VOD