INTERNATIONAL STANDARD

ISO/IEC 29116-1

First edition
2008-06-01

Information technology — Supplemental media technologies —

Part 1: Media streaming application format protocols

Technologies de l'information — Technologies de milieux supplémentaires —

Partie 1: Protocoles de format d'application de diffusion de milieux

Reference number
ISO/IEC 29116-1:2008(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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

ISO/IEC 29116-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 29116 consists of the following parts, under the general title *Information technology — Supplemental media technologies*:

— *Part 1: Media streaming application format protocols*

Introduction

ISO/IEC 29116 is a family of International Standards that has been developed for the purpose of providing the complete line-up of standards that are required to practically deploy Multimedia Application Format standards. The parts of ISO/IEC 29116 have been developed starting from submissions received by proponents. The proposed technologies have been thoroughly reviewed prior to submission of the Committee Draft and have undergone the full national body review during the process of balloting the Draft International Standards.

Information technology — Supplemental media technologies —

Part 1:
Media streaming application format protocols

1 Scope

This International Standard specifies a set of protocols to be used in conjunction with ISO/IEC 23000-5 (Media streaming Player) in applications where governed audio and video information is streamed to an end-user device.

This International Standard specifies two types of protocols,

— Access Protocols, allowing a device to obtain from another device a content item or parts thereof, a license, or executable code implementing security functions, and

— Domain Management Protocols, allowing a number of devices to create, join, administer, etc. a group of users and devices where the participants share common properties.

2 Normative references

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

ISO/IEC 23000-5, *Information technology — Multimedia application format (MPEG-A) — Part 5: Media streaming application format*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
Content Provider Device
device delivering content to another **device**

3.2
Device
combination of hardware and software or just an instance of software that allows a **user** to perform actions

3.3
Domain Administrator
user creating and administering a domain by means of a **Domain Management Device**

3.4
Domain Management Device
device managing the lifecycle of a domain and the membership of **devices** and **users** part of it.

3.5
IPMP Processor
module in a Media Streaming Player in charge of retrieving, instantiating, initialising and managing the **IPMP Tools** required to perform actions on content.

3.6
IPMP Tool
module performing (one or more) IPMP functions such as authentication, decryption, watermarking, etc.

3.7
IPMP Tool Agent
module instantiating, initialising, authenticating, and supervising any operation performed between IPMP Tools within an **IPMP Tool Group**

3.8
IPMP Tool Body
executable code implementing either a Single IPMP Tool or an **IPMP Tool Pack**

3.9
IPMP Tool Group
combination of several IPMP Tools

3.10
IPMP Tool Pack
module that comprises an IPMP Tool Group and its **IPMP Tool Agent**

3.11
IPMP Tool Provider Device
device delivering IPMP Tools to another **device**

3.12
Licence Provider Device
device delivering licenses to another **device**

3.13
User
any identified entity interacting in a media streaming environment using a media streaming **device**.

4 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

CPD	Content Provider Device
DID	Digital Item Declaration
DIDL	Digital Item Declaration Language
DII	Digital Item Identification
DMD	Domain Management Device
IPMP	Intellectual Property Management and Protection
LPD	License Provider Device
LLAP	Local License Access Protocol

MSD Media Streaming Device

MSP Media Streaming Player

RCAP Remote Content Access Protocol

RLAP Remote License Access Protocol

URI Uniform Resource Identifier

5 Namespace conventions

Throughout this part of ISO/IEC 29116, Qualified Names are written with a namespace prefix followed by a colon followed by the local part of the Qualified Name.

For clarity, throughout this part of ISO/IEC 29116, consistent namespace prefixes are used. Table 1 gives these prefixes and the corresponding namespace.

Table 1 — Namespaces and prefixes

Prefix	Corresponding namespace
ipmpdidl	urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS
ipmpmsg	urn:mpeg:mpeg21:2006:07-IPMPMESSAGES-NS
ipmpinfo	urn:mpeg:mpeg21:2004:01-IPMPINFO-NS
didl	urn:mpeg:mpeg21:2002:02-DIDL-NS
didmodel	urn:mpeg:mpeg21:2002:02-DIDMODEL-NS
didl-msx	urn:mpeg:maf:schema:mediastreaming:DIDLextensions
dii	urn:mpeg:mpeg21:2002:01-DII-NS
r	urn:mpeg:mpeg21:2003:01-REL-R-NS
sx	urn:mpeg:mpeg21:2003:01-REL-SX-NS
m1x	urn:mpeg:mpeg21:2005:01-REL-M1X-NS
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance
dsig	http://www.w3.org/2000/09/xmldsig#
msap	urn:mpeg:maf:schema:mediastreaming:accessprotocol:2007
msd	urn:mpeg:maf:schema:mediastreaming:domain:2007
msdp	urn:mpeg:maf:schema:mediastreaming:domainprotocol:2007
msbp	urn:mpeg:maf:schema:mediastreaming:baseprotocol:2007

6 System overview

ISO/IEC 29116-1 specifies the format of the data exchanged between a Media Streaming Player and other Media Streaming Devices, namely:

a. *Content Provider Device,* a device capable of interacting with a *Media Streaming Player* to provide Media Streaming Content
b. *Licence Provider Device,* a device capable of interacting with a *Media Streaming Player* to provide Licences
c. *IPMP Tool Provider Device,* a device capable of interacting with a *Media Streaming Player* to provide IPMP Tools[1)]
d. *Domain Management Device,* a device capable of managing various functions needed for a proper functioning of a domain.

7 Access Protocols

7.1 Introduction

This section specifies the messages exchanged between devices when communicating with the purpose of obtaining from another device:

Content
A license
An IPMP Tool Body

7.2 Base Protocol Representation

This section specifies the base information commonly used in both the access protocols and the domain management protocols.The namespace msbp defines the elements on which the access protocols and the domain protocols are based.

7.2.1 ProtocolBaseType

The msbp:ProtocolBaseType abstract complex type is defined in the figure below. All the complex types defined in this standard extends msbp:ProtocolBaseType.

```
<complexType name="ProtocolBaseType" abstract="true"/>
```

Figure 1 — The msbp:ProtocolBaseType complex type

7.2.2 ProtocolType

The abstract msbp:ProtocolType complex type, defined in the figure below, extends the msbp:ProtocolBaseType for conveying the msbp:TransactionID element which conveys a value which is used to track a message exchange session. Any message in response to another message shall specify the same TransactionID value contained in the request.

```
<complexType name="ProtocolType" abstract="true"/>
   <complexContent>
      <extension base="msbp:ProtocolBaseType">
         <sequence>
            <element name="TransactionID" type="string"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 2 — The msbp:ProtocolType complex type

1) Related terms include: IPMP Processor, IPMP Tool Agent, IPMP Tool Group, IPMP Tool Pack.

7.2.3 Ack

The msbp:Ack element defined in the figure below extends the msbp:ProtocolType complex type by specifying a boolean attribute, Result, which shall indicate whether the protocol was carried out with success or otherwise, and the msbp:ProtocolResult element, that may convey further information concerning the result of an operation.

```
<element name="Ack" type="msbp:AckType"/>
<complexType name="AckType">
   <complexContent>
      <extension base="msbp:ProtocolType">
         <sequence minOccurs="0">
            <element ref="msbp:ProtocolResult"/>
         </sequence>
         <attribute name="Result" type="boolean" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

Figure 3 — The msbp:Ack element

7.2.4 ProtocolResult

The ProtocolResult element may convey either one of the codes specified in Table 2, of a user-defined result code. Furthermore, the DisplayString element may convey a string to be shown to a user as the result of the operation.

```
<element name="ProtocolResult" type="msbp:ProtocolResultType"/>
   <complexType name="ProtocolResultType ">
      <complexContent>
         <extension base="msbp:ProtocolBaseType">
            <sequence>
               <choice>
                  <element name="ResultCode" type="msbp:ResultCodeType"/>
                  <element name="UserDefinedResult" type="string"/>
               </choice>
               <element name="DisplayString" type="string" minOccurs="0"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <simpleType name="ResultCodeType">
      <restriction base="hexBinary">
         <enumeration value="00"/>
         <enumeration value="01"/>
         <enumeration value="02"/>
         <enumeration value="03"/>
         <enumeration value="04"/>
         <enumeration value="05"/>
         <enumeration value="06"/>
         <enumeration value="07"/>
      </restriction>
   </simpleType>
</complexType>
```

Figure 4 — The msbp:ProtocolResult element

A list of result codes is given in the table below.

Table 2 — List of Result Code

Result Code	Description
"00"	RESERVED
"01"	OK
"02"	UNKNOWN_MESSAGE
"03"	TIMEOUT
"04"	UNABLE_TO_PROCESS
"05"	UNKNOWN_FAILURE
"06"	PERMISSION_DENIED
"07"	BUSY

7.3 Access Protocols Representation specification

7.3.1 Introduction

The namespace identified by the prefix msap, indicates protocols to access content, licenses and keys.

7.3.2 AccessProtocolType

The msap:AccessProtocolType complex type, defined in the figure below, extends the msbp:ProtocolType.

```
<complexType name="AccessProtocolType" abstract="true">
   <complexContent>
      <extension base="msbp:ProtocolType"/>
   </complexContent>
</complexType>
```

Figure 5 — The msap: ProtocolType complex type

7.3.3 Ack

The msap:Ack element defined in the figure below extends the msap:AccessProtocolType complex type by specifying a boolean attribute, Result, indicating whether the protocol was carried out with success or otherwise, and the msbp:ProtocolResult element, that may convey further information concerning the result of an operation.

```
<element name="Ack" type="msap:AckType"/>
<complexType name="AckType">
   <complexContent>
      <extension base="msap:AccessProtocolType">
         <sequence minOccurs="0">
            <element ref="msbp:ProtocolResult"/>
         </sequence>
         <attribute name="Result" type="boolean" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

Figure 6 — The msap:Ack element

7.3.4 RequestContent

A device sends a Request Content message (specified in the figure below) to the content provider device in order to access content.

```
<element name="RequestContent" type="msap:RequestContentType"/>
<complexType name="RequestContentType">
   <complexContent>
      <extension base="msap:AccessProtocolType">
         <sequence>
            <element name="ContentIdentifier"
type="msap:ContentIdentifierType"/>
            <element name="MimeType" type="string" minOccurs="0"/>
            <element ref="r:license" minOccurs="0"/>
            <element name="UsageEnvironmentDescription"
type="dia:UsageEnvironmentType" minOccurs="0"/>
            <element name="UsageEnvironmentDescription" type="string"
minOccurs="0"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 7 — The msap:RequestContent element

The msap:RequestContent message conveys the following information:

- msap:ContentIdentifier: the identifier of the requested content item or content element within a content item, as defined in the figure below
- msap:MimeType: the Mime Type of the content being requested. The following values are permitted:
  - application/mp21 – the MPEG-21 file is requested
  - application/xml – the digital item representing the content item or content element identified in msap:ContentIdentifier is requested
  - the mime type of the resource identified in msap:ContentIdentifier
- r:license: an optional license specifying additional information about the requested license needed to access the requested content
- msap:UsageEnvironmentDescription: Tool for describing the usage environment. Each UsageEnvironmentProperty child element describes a property of the usage environment, such as User characteristics, or terminal capabilities, or network characteristics, or natural environment characteristics
- dsig:Signature: an optional digital signature of the msap:RequestContent message by the device

The ContentIdentifierType complex type specified in the figure below conveys the identifier of a content item and optionally the identifier of a content element within the content item. In the case the msap:ContentElementIdentifier element is specified in an msap:RequestContent message, this implies that only the specific content element is requested, and not the whole content item.

```
<complexType name="ContentIdentifierType">
   <complexContent>
      <extension base="msbp:ProtocolBaseType">
         <sequence>
            <element name="ContentItemIdentifier" type="anyURI"/>
            <element name="ContentElementIdentifier" type="anyURI"
minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 8 — The msap:ContentIdentifierType complex type

7.3.5 RequestContentResponse

The RequestContentResponse message, sent in response to a RequestContent message, is specified in the figure below.

```
<element name="RequestContentResponse" type="msap:RequestContentResponseType"/>
<complexType name="RequestContentResponseType">
   <complexContent>
      <extension base="msap:AccessProtocolType">
         <sequence>
            <element name="DI" type="didl:DIDLType" minOccurs="0"/>
            <element name="ContentURL" type="msap:ContentURLType" minOccurs="0"
maxOccurs="unbounded"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 9 — The msap:RequestContentResponse element

The msap:RequestContentResponse message is employed by the Content Provider Device to deliver the following information:

- msap:DI: the optional digital item representing the content item being requested
- msap:ContentURL: an optional sequence of elements (whose syntax is specified in the figure below) specifying the URLs from where the requested resource and any associated metadata can be obtained
- dsig:Signature: an optional digital signature applied to the message

7.3.6 ContentURL

If the target of a RequestContent message is a content element part of a content item, depending on the nature of the content element being requested, a number of resources may be made available to the requesting party. The ContentURL complex type allows signalling the mime type of each resource and the URL from which each resource is available. As an example, if the content element being requested consists of a media resource (e.g. an audio elementary stream) and associated metadata, two separate msap:ContentURL elements shall be returned in the msap:RequestContentResponse, one indicating the URL for the audio elementary stream and the other the URL for the metadata elementary stream.

The msap:ContentURLType complex type is specified in the figure below.

```
<complexType name="ContentURLType">
   <complexContent>
      <extension base="msbp:ProtocolBaseType">
         <sequence>
            <element name="MimeType" type="string"/>
            <element name="URL" type="anyURI"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 10 — The msap:RequestContentResponse element

7.3.7 Request License

The msap:RequestLicense specified in the figure below is sent by a device to a License Provider Device in order to request a license granting the device or the user of the device one or more rights over a content item or a content element part of a content item. The msap:RequestLicense message allows requesting a license for either a content item or content element, or a license having a specific license identifier.

```
<element name="RequestLicense" type="msap:RequestLicenseType"/>
<complexType name="RequestLicenseType">
   <complexContent>
      <extension base="msap:AccessProtocolType">
         <sequence>
            <choice>
               <element name="ContentIdentifier"
type="msap:ContentIdentifierType"/>
               <element name="LicenseID" type="anyURI"/>
            </choice>
            <element ref="r:license" minOccurs="0"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 11 — The msap:RequestLicense element

The semantics for the msap:RequestLicense message is given below:

- msap:ContentIdentifier: the identifier of the governed asset for which a license is requested
- msap:LicenseID : The identifier of the license being requested
- r:license: an optional license specifying the principal(s), the right(s), the resource(s) and the condition(s) that the requesting party would like to be specified in the license being requested.
- dsig:Signature: an optional digital Signature of the msap:RequestLicense message.

7.3.8 RequestLicenseResponse

The msap:RequestLicenseResponse message, sent in response to an msap:RequestLicense message, is specified in the figure below.

```
<element name="RequestLicenseResponse" type="msap:RequestLicenseResponseType"/>
<complexType name="RequestLicenseResponseType">
   <complexContent>
      <extension base="msap:AccessProtocolType">
         <sequence>
            <element ref="r:license"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 12 — The msap:RequestLicenseResponse element

The msap:RequestLicenseResult element is employed by the License Provider to deliver a license, which shall be included in the r:license element. Shall this message be signed, the digital signature shall be included in the dsig:Signature element.

7.3.9 RequestIPMPToolBody

The msap:RequestIPMPToolBody message, specified in the figure below, is employed by a device to request an IPMP Tool Body to an IPMP Tool Provider Device.

```
<element name="RequestIPMPToolBody" type="msap:RequestIPMPToolBodyType"/>
<complexType name="RequestIPMPToolBodyType">
   <complexContent>
      <extension base="msap:AccessProtocolType">
         <sequence>
            <element ref="ipmpinfo:IPMPToolID"/>
            <element ref="ipmpinfo-msx:DeviceInformation"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 13 — The msap:RequestIPMPToolBody element

The msap:RequestIPMPToolBody element conveys the following information:

- ipmpinfo:IPMPToolID: the identifier of the IPMP Tool whose IPMP Tool Body is requested
- ipmpinfo-msx:DeviceInformation: information about the hardware and/or software characteristics of the device on which the requested IPMP Tool Body shall operate.

7.3.10 RequestIPMPToolBodyResponse

The msap:RequestIPMPToolBodyResponse message, sent in response to an msap:RequestIPMPToolBody message, is specified in the figure below.

```
<element name="RequestIPMPToolBodyResponse"
type="msap:RequestIPMPToolBodyResponseType"/>
<complexType name="RequestIPMPToolBodyResponseType">
   <complexContent>
      <extension base="msap:AccessProtocolType">
         <sequence>
            <choice maxOccurs="unbounded">
               <element ref="ipmpinfo-msx:ToolBody"/>
               <element name="ToolURL" type="anyURI"/>
            </choice>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 14 — The msap:RequestIPMPToolBodyResponse element

The msap:RequestIPMPToolBodyResponse element is employed by an IPMP Tool Provider Device to deliver an IPMP Tool Body. This message may contain one or more Tool Body elements conveyed within the element ipmpinfo-msx:ToolBody, or may specify a number of remote locations from where the Tool Body elements can be obtained.

7.4 Access Protocol specification

This section specifies how to employ the messages defined in 7.3 to request content items or parts thereof, licenses and IPMP Tool Bodies. The messages exchanged between two devices are based on a transactional protocol called Remote Access Protocol (RAP) that is supported over an existing network protocol (TCP/IP or HTTP in the case of Internet/WWW access). In terms of security, the RAP uses two security layers:

- Application-Level: this corresponds to the messages described in the RAP in this document;
- Network-Level: this is represented by an underlying security protocol, i.e. the SSLv3/TLSv1 protocol.

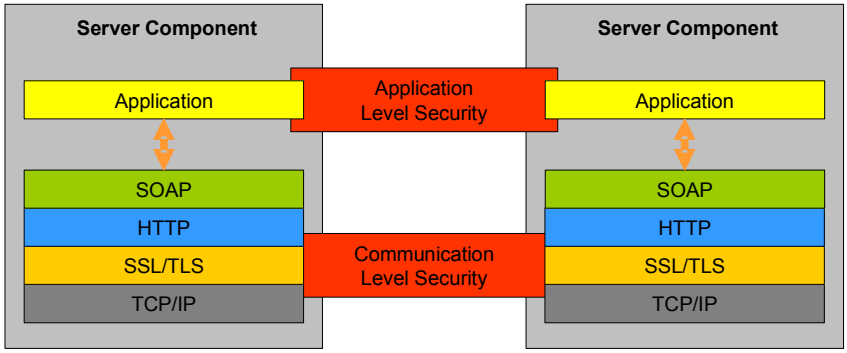

Figure 15 — RAP Security Layers

7.4.1 Access Content Protocol specification

This section specifies the Access Content Protocol. Successful execution of the protocol may results in content being available on the requesting device. This protocol is based on the exchange of messages between two basic components: the requesting device and the Content Provider Device.

The RCAP involves the following steps:

a) The requesting device and the Content Provider Device mutually authenticate
b) The requesting device:
    1) generates a msap:RequestContent message [Figure 7].
    2) optionally signs the message.
    3) sends the msap:RequestContent message to the Content Provider Device;
c) The Content Provider Device, upon receiving the message:
    1) verifies the digital signature if this is present
    2) verifies that the content ID and the content element ID (the latter only if present), specified in the message are valid and available in the Content Provider Device database/file system
    3) determines on the bases of the Mime Type parameter whether the target of the request is the whole content item as a digital item or as an MPEG-21 File, or is a specific content element part of the content item
    4) determines whether the UsageEnvironmentDescription parameter contains a valid Usage Environment Description and the adaptation requested is possible, the requested content item or the specific resource is adapted.
d) In the case the request can be satisfied, the Content Provider Device generates and conveys to the requesting Device an msap:RequestContentResponse message [Figure 9] containing:
    1) the digital item representing the requested content if the msap:MimeType element in the request contained the value "application/xml"
    2) one or more URLs from where the whole content item can be retrieved, in the case the msap:MimeType element in the request contained the value "application/mp21"
    3) one or more URLs specifying the location from where the content element identified by the msap:ContentElementID parameter in the request can be retrieved, in the case the msap:ContentElementID parameter in the request was specified
    4) (optionally) the digital signature for the response message
e) In the case the request cannot be satisfied, the Content Provider Device generates and conveys to the requesting device an msap:Ack message [Figure 6] conveying the information related to the reason of failure.
f) The requesting Device:
    1) (optionally) replies with a msap:Ack message [Figure 6]
    2) retrieves the content from the location specified in the msap:RequestContentResult message and uses it according to the license terms

7.4.2 Access License Protocol specification

The Remote License Access Protocol (RLAP) is performed by a device to access licenses from a License Provider Device. At the application level, RLAP is based on the exchange of messages between two basic components: the requesting device and the License Provider Device.

The RLAP involves the following steps:

a) The requesting device:
   1) verifies that the content item or content element is protected and that a license needed to access the item of content is not available
   2) extracts the content item or content element identifier from the digital item
   3) retrieves the ipmpinfo:LicenseReference URL from the ipmpinfo:RightsDescriptor element in the digital item
b) The requesting device and the License Provider Device mutually authenticate;
c) The requesting device generates an msap:RequestLicense message [Figure 11] containing the following elements:
   1) a choice between either (i) the content identifier or the content element identifier, or (ii) the License Identifier (the latter in the case the identifier of a specific license that would grant the needed rights is known)
   2) (optionally) the license that the requesting device would require in order to use the governed content item or content element
d) The requesting device
   1) (optionally) signs the message
   2) sends the msap:RequestLicense message to the License Provider Device
e) The License Provider Device, upon receiving the message:
   1) verifies the signature (if present)
   2) verifies that either (i) a valid content item or content element identifier are specified in the request (in this case, the optional license element, if present, indicates the type of license requested) or (ii) a valid license identifier is specified in the request
   3) verifies whether (i) there is a license on the system, or (ii) if the requesting device is entitled to be issued a license for the specific content item or content element identifier
f) In the case the request can be satisfied, the License Provider Device:
   1) generates a msap:RequesLicenseResponse message [Figure 12] containing the license. In the case the license grants rights over a protected resource, the license shall include the decryption key needed by the requesting device or user to decrypt the key(s) contained in the digital item which is (are) needed to decrypt the protected resource.
   2) delivers the msap:RequesLicenseResult message to the requesting Device:
g) In the case the request cannot be satisfied, the License Provider Device generates and sends to the requesting device an msap:Ack message [Figure 6] to acknowledge the failure of the protocol:
h) The requesting device receives the message from the License Provider Device, verifies the message contents and in case of success extracts the license;
i) (optionally) the requesting device sends a msap:Ack message [Figure 6] to the License Provider Device signalling that the response was successfully received.

7.4.3 Access IPMP Tool Body Protocol specification

This protocol is performed by a Media Streaming Player to retrieve an IPMP Tool Body from an IPMP Tool Provider Device. This protocol is carried out in the following steps:

a) The Media Streaming Player mutually authenticates with the IPMP Tool Provider Device. The URL of the IPMP Tool Provider Device is specified in the ipmpinfo:Remote element in the DI
b) The Media Streaming Player sends an msap:RequestIPMPToolBody message [Figure 13] to the IPMP Tool Provider Device, containing:
   1) the identifier of requested IPMP Tool
   2) the ipmpinfo-msx:DeviceInformation structure specifying the hardware and software characteristics of the device on which the IPMP Tool shall operate
c) The IPMP Tool Provider Device, upon receiving the message, performs the following operations:
   1) verifies the digital signature of the message if present
   2) reads the ipmpinfo-msx:DeviceInformation contained in the request message

3) locates the IPMP Tool Body matching the requesting device's characteristics contained within DeviceInformation
4) wraps the IPMP Tool Body in an ipmpinfo-msx:ToolBody structure
5) delivers the msap:RequestIPMPToolBodyResponseToolBody [Figure 14] to the Media Streaming Player

d) The Media Streaming Player
1) receives the msap:RequestIPMPToolBodyResponse message
2) (optionally) sends an msap:Ack message to the IPMP Tool Provider Device to notify the receipt (or otherwise) of the IPMP Tool Body
3) reads the ipmpinfo-msx:ToolPackageType information and unpackages the ipmpinfo-msx:ToolBody
4) stores the IPMP Tool Body in the local storage

8 Domain management protocols

8.1 Introduction

This section specifies the protocols between an end-user device and devices involved in managing a domain.

8.2 Domain management overview

Domains are groups of devices or users sharing some common properties. By using domains it becomes possible to implement more flexible licensing modalities, e.g. to license content to all devices or users in a domain. A domain is managed by a special device called the Domain Management Device, and it is administered by a Domain Administrator.

Figure 16 represents a possible domain configuration with a Domain Management Device, a License Provider Device, and a number of end-user devices and their users which may or may not belong to the same Domain managed by the Domain Management Device.

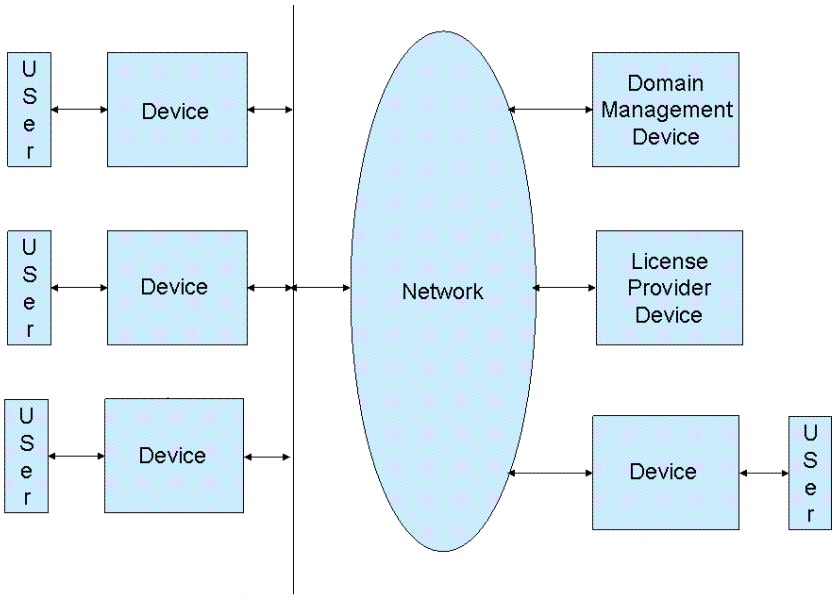

Figure 16 — An example of domain

In general to manage a domain 5 device types are required:

1. **Domain Management Devices** (DMD), to manage the life cycle of domains and the list of devices and users belonging to the domains;
2. **Domain Identification Devices** (DoID), to assign Globally Unique Identifiers (GUID) to DMDs on behalf of appropriate registration authorities;
3. **End User Devices** (EUD);
4. **Users,** bearing in mind that users are represented by e.g. a device (smart card etc.) or an identity on a device (UN/PW etc.);
5. **License Provider Devices** (LPD) to provide licenses including for use of content in a domain.

The combined operation of these devices can be shown by a simple walkthrough in the figure below.

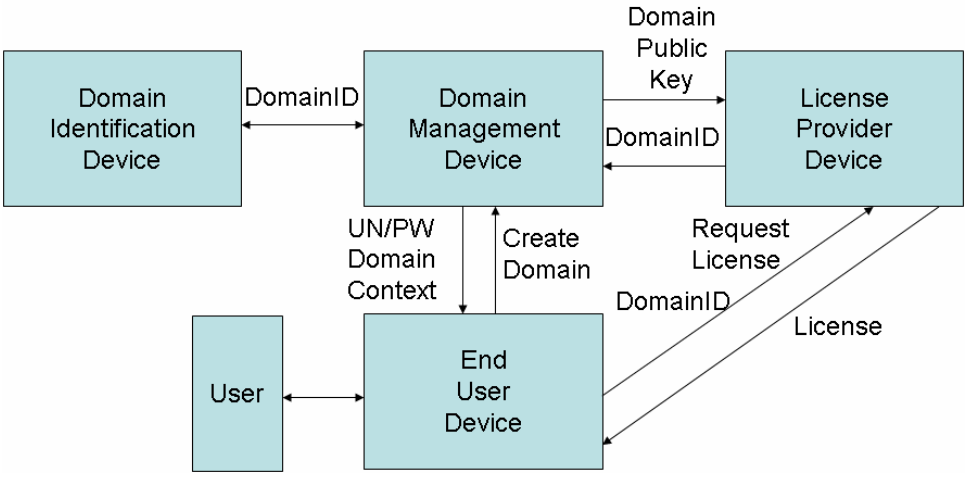

Figure 17 — Some of the relationship of the 5 Device types in Manage Domain

The following walkthrough describes some operations occurring when creating a domain or within a domain:

a) The Domain Administrator requests to the DMD to create a domain, specifying additional information such as the list of device/users, the maximum number of devices/users permitted in the domain, etc.
b) The DMD
    1) obtains an identifier for the domain from a Domain Identification Device
    2) delivers domain credentials to the Domain Administrator
c) Devices or users may request to the DMD the membership to the domain
d) The DMD issues the license granting the membership to the domain to the device/user
e) Devices or users may renew their membership to the domain when this expires, or they may leave a domain

Note A Domain Management Device can manage one or more domains at the same time. Ownership of a DMD can be implemented using a variety of mechanisms, e.g. end-user based or service provider based.

8.3 Domain Information specification

This section specifies the information associated with domain management.

8.3.1 Common elements defined in the msd namespace

The domain namespace defines the elements in the figure below.

```
<element name="DomainManagerID" type="r:KeyHolder"/>
<element name="AccessPassword" type="string"/>
<element name="AccessID" type="string"/>
<element name="DomainKey" type="xenc:EncryptedKeyType"/>
<element name="UserID" type="msd:IDType"/>
```

```
<element name="DeviceID" type="msd:IDType"/>
<element name="LocalDomainID" type="msd:IDType"/>
<element name="ContentGroupID" type="anyURI"/>
<element name="MaximumNumberOfDevices" type="unsignedInt"/>
<element name="MaximumNumberOfUsers" type="unsignedInt"/>
<element name="MaximumFrequencyOfUpdateDevice" type="duration"/>
<element name="MaximumFrequencyOfUpdateUser" type="duration"/>
<element name="Expiration" type="sx:ValidityTimeMetered"/>
```

Figure 18 — Common msd elements

The semantics for the elements in the figure above are given below:

- `msd:DomainManagerID`: the identifier assigned to a Domain Manager Device (DMD) by an appropriate registration authority
- `msd:AccessID`: the Access ID given to the Domain Administrator
- `msd:AccessPassword`: the Access Password given to the Domain Administrator
- `msd:DomainKey`: the decryption key for Domain-encrypted content key.
- `msd:UserID`: the identifier associated with a user, of type msd:IDType (See 8.3.3)
- `msd:DeviceID`: the Identifier associated with a device, of type msd:IDType (See 8.3.3)
- `msd:LocalDomainID`: the Local Domain Identifier issued by the Domain Identification Device
- `msd:ContentGroupID`: the Identifier assigned to a group of content items for the management of domains
- `msd:MaximumNumberOfDevices`: the maximum number of devices allowed in a domain
- `msd:MaximumNumberOfUsers`: the maximum number of users allowed in a domain
- `msd:MaximumFrequencyOfUpdateDevice`: the minimum time interval before a device that has left a domain may be allowed to re-join it
- `msd:MaximumFrequencyOfUpdateUser`: the minimum time interval before a user who has left a domain may be allowed to re-join it
- `msd:Expiration` : The time at which the domain expires

8.3.2 DomainBaseType

The Media Streaming Domain namespace, msd, defines an abstract complex type from which a number of complex types defined in the same namespace are derived. This is defined as follows:

```
<complexType name="DomainBaseType" abstract="true"/>
```

Figure 19 — The msd:DomainBaseType complex type

8.3.3 IDType

The msd:IDType is a complex type conveying the identifier of either a user or a device.

```
<complexType name="IDType">
   <sequence>
      <choice>
         <element name="id" type="anyURI"/>
         <element ref="dsig:X509Data" minOccurs="0"/>
      </choice>
   </sequence>
</complexType>
```

Figure 20 — The msd:IDType complex type

The msd:IDType complex type contains an identifier for either a device or a user in a domain, which can be either of type anyURI and conveyed by the msd:id element, or an X.509 certificate, in which case it shall be expressed according to the dsig:X509Data element and conformant to [13].

8.3.4 DomainManageInfo

A domain is created when a Domain Administrator requests the Domain Manager Device the creation of a new domain. The protocol for creating a domain is described in 8.6.2. The result of this protocol is the creation by the DMD of a msd:DomainManageInfo element specified in the figure below.

```
<element name="DomainManageInfo" type="msd:DomainManageInfoType"/>
<complexType name="DomainManageInfoType">
   <complexContent>
      <extension base="msd:DomainBaseType">
         <sequence>
            <element ref="msd:DomainID"/>
            <element ref="msd:DACredentials" minOccurs="0"/>
            <element ref="msd:DomainMembershipCredentials" minOccurs="0"/>
            <choice minOccurs="0" maxOccurs="2">
               <element ref="msd:User"/>
               <element ref="msd:Device"/>
            </choice>
            <element ref="msd:DomainKey"/>
            <element name="Registration" type="dateTime"/>
            <element ref="msd:Expiration"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 21 — The msd:DomainManageInfo element

In the above, the following semantics apply:

- the msd:Registration element indicates the time at which the domain is created.
- the msd:DACredentials and msd:DomainMembershipCredentials elements indicate credential information required for domain access and management of domain membership, as defined below.

8.3.5 DACredentials and DomainMembershipCredentials

The msd:DACredentials element conveys the Domain Administration credentials which a Domain Administrator shall possess in order to perform any operation on a domain by means of a DMD. The msd:DomainMembershipCredentials are credentials employed by a device or a user to authenticate with a DMD.

```
<element name = "DACredentials" type="msd:DomainCredentialType"/>
<element name = "DomainMembershipCredentials" type="msd:DomainCredentialType"/>
<complexType name="DomainCredentialType ">
   <sequence>
      <choice>
         <element ref="msd:AccessID"/>
         <element ref="msd:AccessPassword"/>
      </choice>
   </sequence>
</complexType>
```

Figure 22 — The msd:DACredentials and msd:DomainMembershipCredentials elements

8.3.6 DomainID

The msd:DomainID element is defined in the figure below.

```
<element name="DomainID" type="msd:DomainIDType"/>
<complexType name="DomainIDType">
   <complexContent>
      <extension base="msd:IDType">
         <sequence>
            <element ref="msd:DomainManagerID"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 23 — The msd:DomainID element

The DomainID element shown above extends the msd:IDType by conveying the msd:DomainManagerID element, the identifier assigned to the Domain Management Device. The DomainID element is comprised of the msd:LocalDomainID issued by the Domain Identification Device, and the msd:DomainManagerID.

8.3.7 User

The msd:User element conveys a set of properties associated with users in a domain. The msd:User element is defined in the figure below.

```
<element name="User" type="msd:UserType"/>
<complexType name="UserType">
   <complexContent>
      <extension base="msd:DomainBaseType">
         <sequence>
            <element ref="msd:UserIDList"/>
            <element ref="msd:MaximumNumberOfUsers" minOccurs="0"/>
            <element ref="msd:MaximumFrequencyOfUpdateUser" minOccurs="0"/>
            <element ref="msd:UserRevocationList" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 24 — The msd:User element

The msd:User element conveys a set of properties associated with Users in a Domain. This conveys the following information:

- the list of all users in the domain, msd:UserIDList, as defined in the figure below
- the msd:UserRevocationList: the list of users which are no longer allowed to be members of this domain

Note The UserID shall not be removed from the UserIDList until the duration time indicated in MaximumFrequencyOfUpdateUser has passed.

The msd:UserIDList element is defined in the Figure below.

```
<element name="UserIDList" type="msd:UserIDListType"/>
<complexType name="UserIDListType">
   <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="msd:UserID"/>
      <element ref="msd:Expiration"/>
   </sequence>
</complexType>
```

Figure 25 — The msd:UserIDList element

This element is used to convey a list of user identifiers associated with this domain. Each user has an expiration time, allocated by the DMD at the time of joining the domain.

8.3.8 Device

The msd:Device element conveys a set of properties associated with devices in a domain. The msd:Device element is defined in the figure below.

```
<element name="Device" type="msd:DeviceType"/>
<complexType name="DeviceType">
   <complexContent>
      <extension base="msd:DomainBaseType">
         <sequence>
            <element ref="msd:DeviceIDList"/>
            <element ref="msd:MaximumNumberOfDevices" minOccurs="0"/>
            <element ref="msd:MaximumFrequencyOfUpdateDevice" minOccurs="0"/>
            <element ref="msd:DeviceRevocationList" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 26 — The msd:Device element

The msd:Device element shown above conveys the following set of properties associated with devices in a domain:

- the msd:DeviceIDList, as defined in the Figure below
- the msd:DeviceRevocationList: the list of devices which are no longer allowed to be members of this domain

The msd:DeviceIDList element is defined in the figure below.

```
<element name="DeviceIDList" type="msd:DeviceIDListType"/>
<complexType name="DeviceIDListType">
   <sequence minOccurs="0" maxOccurs="unbounded">
      <element ref="msd:DeviceID"/>
      <element ref="msd:Expiration"/>
   </sequence>
</complexType>
```

Figure 27 — The msd:DeviceIDList element

The msd:DeviceIDList element is used to convey a list of device identifiers associated with this domain. Each device has an expiration time, allocated by the DMD at the time of joining the domain.

8.4 Domain Use Data specification

The representation of use data required for the management of domains is given below. The way msd:UseData and msd:Record elements are employed is described in this section.

8.4.1 UseData

The msd:UseData element is employed to collect a number of msd:Record elements and the identifier of the domain to which the record refers to.

```
<element name="UseData" type="msd:UseDataType"/>
<complexType name="UseDataType">
   <sequence>
```

```
        <element ref="msd:DomainID"/>
        <element ref="msd:Record" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

Figure 28 — The msdp:UseData element

8.4.2 Record

The `msdp:Record` element is given in the figure below.

```
<element name="Record" type="msd:RecordType"/>
<complexType name="RecordType">
    <sequence>
        <element ref="msd:DeviceID"/>
        <element name="StartTime" type="dateTime"/>
        <element name="EndTime" type="dateTime"/>
        <element name="NumberOfContentGroups" type="integer"/>
        <element ref="msd:ContentGroupID" minOccurs="0" maxOccurs="unbounded"/>
        <element name="NotificationFlag" type="boolean"/>
    </sequence>
</complexType>
```

Figure 29 — The msdp:Record element

The semantics for `msdp:Record` is given below:

- DeviceID: the identifier of the device on which the content was used
- UserID: the identifier of the user by which the content was used
- StartTime: the time the device started to use an item of content belonging to a Content Group
- EndTime: the time the device ended using an item of content belonging to a Content Group
- NumberOfContentGroups: the number of Content Groups to which the item of content being used belongs to. Each Content Group may consist of multiple items of content, where only one of them in a Content Group is allowed to be used simultaneously.
- ContentGroupID: The Content Group identifiers
- NotificationFlag: a boolean value set to TRUE when this Use Data record has been notified to the DMD. The communication of the Use Data to another device in the domain doesn't change the value of this flag in the Use Data.

8.5 Domain Protocol Information specification

This section specifies the messages exchanged between entities in a domain, for instance when creating a domain, or when devices join or leave a domain. The messages defined in this section include a number of elements defined in the msd namespace specified in 8.3, and new elements in the Media Streaming Domain Protocol namespace, msdp.

8.5.1 DomainProtocolType

The msdp:DomainProtocolType complex type, defined in the figure below, extends the msbp:ProtocolType.

```
<complexType name="DomainProtocolType" abstract="true">
    <complexContent>
        <extension base="msbp:ProtocolType"/>
    </complexContent>
</complexType>
```

Figure 30 — The msdp: DomainProtocolType complex type

8.5.2 Ack

The msdp:Ack element defined in the figure below extends the msdp:DomainProtocolType complex type by specifying a boolean attribute, Result, indicating whether the protocol was carried out with success or otherwise, and the msbp:ProtocolResult element, that may convey further information concerning the result of an operation.

```
<element name="Ack" type="msdp:AckType"/>
<complexType name="AckType">
   <complexContent>
      <extension base="msdp:DomainProtocolType">
         <sequence minOccurs="0">
            <element ref="msbp:ProtocolResult"/>
         </sequence>
         <attribute name="Result" type="boolean" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

Figure 31 — The msdp:Ack element

8.5.3 AuthenticateReq

The msdp:AuthenticateReq message is sent by a device or a user to a DMD in order to request authentication.

```
<element name="AuthenticateReq" type="msdp:AuthenticateReqType"/>
<complexType name="AuthenticateReqType">
   <complexContent>
      <extension base="msbp:ProtocolType">
         <sequence>
            <element ref="msd:DomainID" minOccurs="0"/>
            <choice>
               <element ref="msd:DACredentials"/>
               <element ref="msd:DomainMembershipCredentials"/>
            </choice>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 32 — The msdp:AuthenticateReq element

Depending on whether the sender of the message is a Domain Administrator or a domain member (device or user), DACredientials or DomainMembershipCredentials will be used respectively.

8.5.4 LocalDomainIDRequest

The msdp:LocalDomainIDRequest message specified in the figure below is employed by a DMD to request an identifier for a new domain to a Domain Identification Device.

```
<element name="LocalDomainIDRequest" type="msdp:RequestLocalDomainIDType"/>
<complexType name="RequestLocalDomainIDType">
   <complexContent>
      <extension base="msdp:DomainProtocolType"/>
   </complexContent>
</complexType>
```

Figure 33 — The msdp:LocalDomainIDRequest element

8.5.5 LocalDomainIDResponse

The msdp:LocalDomainIDResponse message specified in the figure below is sent by a Domain Identification Device to a DMD conveying the identifier of a new domain.

```
<element name="LocalDomainIDResponse" type="msdp:LocalDomainIDResponseType"/>
<complexType name="LocalDomainIDResponseType">
   <complexContent>
      <extension base="msdp:DomainProtocolType">
         <sequence>
            <element ref="msd:LocalDomainID"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 34 — The msdp:LocalDomainIDResponse element

8.5.6 RequestKey

The msdp:RequestKey message shown above is sent a License Provider Device to a Domain Management Device to request the domain encryption key so that domain-bound licenses can be issues.

```
<element name="RequestKey" type="msdp:RequestKeyType"/>
<complexType name="RequestKeyType">
   <complexContent>
      <extension base="msdp:DomainProtocolType">
         <sequence>
            <element ref="msd:DomainID"/>
            <element ref="msd:ContentGroupID" minOccurs="0"
maxOccurs="unbounded"/>
            <choice minOccurs="0" maxOccurs="unbounded">
               <element ref="msd:DeviceID"/>
               <element ref="msd:UserID"/>
            </choice>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 35 — The msdp:RequestKey element

The semantics for the msdp:RequestKey message is given below:

- DomainID: the identifier of the domain for which the License Provider Device is requesting the domain key
- ContentGroupID: the list of content groups (group of item of contents for management within domains) licensed to the domain for which the LPD issues a license. A content group identifier is indicated in a license as a resource URI in the m2x:isPartOf element in the m2x:simultaneousAccess condition.
- DeviceID: the identifier of the device requesting a domain-bound license.
- UserID: the identifier of the user requesting a domain-bound license

8.5.7 RequestKeyResponse

The msdp:RequestKeyResponse message specified in the figure below is sent by a DMD to an LPD in response to an msdp:RequestKey message.

```
<element name="RequestKeyResponse" type="msdp:RequestKeyResponseType"/>
<complexType name="RequestKeyResponseType">
```

```
    <complexContent>
       <extension base="msdp:DomainProtocolType">
          <sequence>
             <element ref="msd:DomainKey"/>
             <element ref="msd:UserID" minOccurs="0" maxOccurs="unbounded"/>
             <element ref="msd:DeviceID" minOccurs="0" maxOccurs="unbounded"/>
          </sequence>
       </extension>
    </complexContent>
</complexType>
```

Figure 36 — The msdp:RequestKeyResponse element

The semantics for msdp:RequestKeyResponse message is given below:

- msd:DomainKey: element used to convey the encryption key of the domain indicated by DomainID in the preceding msdp:RequestKey message. This is used subsequently by the License Provider Device to issue licenses for domain members, so that only those devices/users within the domain can use protected content.
- msd:DeviceID: the identifiers of the devices members of the domain identified by the msd:DomainID specified in the preceding msdp:RequestKey.
- msd:UserID: the identifiers of the users members of the domain indicated by msd:DomainID specified in the preceding msdp:RequestKey.

8.5.8 AddDevice

The msdp:AddDevice message specified in the figure below is sent by a Media Streaming Player to a DMD requesting to join a domain.

```
<element name="AddDevice" type="msdp:AddDeviceType"/>
<complexType name="AddDeviceType">
    <complexContent>
       <extension base="msdp:DomainProtocolType">
          <sequence>
             <element ref="msd:DeviceID"/>
             <element ref="msd:Expiration" minOccurs="0"/>
          </sequence>
       </extension>
    </complexContent>
</complexType>
```

Figure 37 — The msdp:AddDevice element

This message conveys the identifier of the device requesting domain membership and optionally its expiration information.

8.5.9 AddUser

The msdp:AddUser message specified in the figure below is sent by a Media Streaming Player to a DMD requesting its user to become a member of a domain.

```
<element name="AddUser" type="msdp:AddUserType"/>
<complexType name="AddUserType">
    <complexContent>
       <extension base="msdp:DomainProtocolType">
          <sequence>
             <element ref="msd:UserID"/>
             <element ref="msd:Expiration" minOccurs="0"/>
          </sequence>
```

```
        </extension>
    </complexContent>
</complexType>
```

Figure 38 — The msdp:AddUser element

This message conveys the identifier of the user requesting domain membership and optionally its expiration information.

8.5.10 RenewDevice

The msdp:RenewDevice message specified in the figure below is sent by a Media Streaming Player to a DMD requesting its membership to the domain to be renewed.

```
<element name="RenewDevice" type="msdp:RenewDeviceType"/>
<complexType name="RenewDeviceType">
    <complexContent>
        <extension base="msdp:DomainProtocolType">
            <sequence>
                <element ref="msd:DeviceID"/>
                <element ref="msd:UseData" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 39 — The msdp:RenewDevice element

This message conveys the identifier of the device requesting the renewal of domain membership, and the use data (see 8.7.2) generated while using content bound to that domain.

8.5.11 RenewUser

The msdp:RenewUser message specified in the figure below is sent by a Media Streaming Player to a DMD requesting the membership of its user to the domain to be renewed.

```
<element name="RenewUser" type="msdp:RenewUserType"/>
<complexType name="RenewUserType">
    <complexContent>
        <extension base="msdp:DomainProtocolType">
            <sequence>
                <element ref="msd:UserID"/>
                <element ref="msd:UseData" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 40 — The msdp:RenewUser element

This message conveys the identifier of the user requesting the renewal of domain membership, and the use data (see 8.7.2) generated while using content bound to that domain.

8.5.12 AddDeviceResponse, AddUserResponse, RenewDeviceResponse and RenewUserResponse

The four messages defined in the figure below are sent in response to 8.5.8, 8.5.9, 8.5.10 and 8.5.11 respectively.

```
<element name="AddDeviceResponse" type="msdp:LicenseResponseType"/>
<element name="AddUserResponse" type="msdp:LicenseResponseType"/>
<element name="RenewDeviceResponse" type="msdp:LicenseResponseType"/>
```

```
<element name="RenewUserResponse" type="msdp:LicenseResponseType"/>
<complexType name="LicenseResponseType">
   <complexContent>
      <extension base="msdp:DomainProtocolType">
         <sequence>
            <element ref="r:license"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 41 — The msdp:AddDeviceResponse, msdp:AddUserResponse, msdp:RenewDeviceResponse, and msdp:RenewUserResponse element

The four messages in the figure above convey a license granting to the user or the device the membership to the domain. An example of domain membership license is provided in the figure below.

```
<license>
   <grant>
      <keyHolder>
         <info>
            <dsig:KeyName>my_device_public_key</dsig:KeyName>
            <dsig:KeyValue>
               <dsig:RSAKeyValue>
                  <dsig:Modulus>01234567</dsig:Modulus>
                  <dsig:Exponent>89ABCDEF</dsig:Exponent>
               </dsig:RSAKeyValue>
            </dsig:KeyValue>
         </info>
      </keyHolder>
      <possessProperty/>
      <m1x:protectedResource>
         <digitalResource>
            <nonSecureIndirect URI="urn:foo:domains:5555"/>
         </digitalResource>
         <xenc:EncryptedKey>
            <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa1024"/>
            <xenc:CipherData>
               <xenc:CipherValue>ABABABAB</xenc:CipherValue>
            </xenc:CipherData>
            <xenc:CarriedKeyName>Domain Key encrypted with
my_device_public_key</xenc:CarriedKeyName>
         </xenc:EncryptedKey>
      </m1x:protectedResource>
      <r:allConditions>
         <validityInterval>
            <notBefore>2007-10-16T00:00:00</notBefore>
            <notAfter>2007-12-31T00:00:00</notAfter>
         </validityInterval>
      </r:allConditions>
   </grant>
   <issuer>
      <keyHolder>
         <info>
            <dsig:KeyName>DMD_Public_Key</dsig:KeyName>
            <dsig:KeyValue>
               <dsig:RSAKeyValue>
                  <dsig:Modulus>12121212</dsig:Modulus>
                  <dsig:Exponent>34343434</dsig:Exponent>
```

```
                </dsig:RSAKeyValue>
            </dsig:KeyValue>
        </info>
    </keyHolder>
</issuer>
</license>
```

Figure 42 — An example of domain membership license

8.5.13 LeaveDevice

The msdp:LeaveDevice message is sent by a Media Streaming Player requesting its membership to the domain to be ceased. The message specified the identifier of the device requesting to be removed from the domain.

```
<element name="LeaveDevice" type="msdp:LeaveDeviceType"/>
<complexType name="LeaveDeviceType">
    <complexContent>
        <extension base="msdp:DomainProtocolType">
            <sequence>
                <element ref="msd:DeviceID"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 43 — The msdp:LeaveDevice element

8.5.14 LeaveUser

The msdp:LeaveUser message is sent by the user of a Media Streaming Player to a DMD requesting the membership of the user to the domain to be ceased. The message specified the identifier of the user requesting to be removed from the domain.

```
<element name="LeaveUser" type="msdp:LeaveUserType"/>
<complexType name="LeaveUserType">
    <complexContent>
        <extension base="msdp:DomainProtocolType">
            <sequence>
                <element ref="msd:UserID"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 44 — The msdp:LeaveUser element

8.5.15 CreateDomain

The msdp:CreateDomain message is sent by a Domain Administrator to a DMD asking the creation of a new domain.

```
<element name="CreateDomain" type="msdp:CreateDomainType"/>
<complexType name="CreateDomainType">
    <complexContent>
        <extension base="msdp:DomainProtocolType">
            <sequence>
                <element ref="msd:DACredentials"/>
                <element ref="msd:Expiration"/>
                <element ref="msd:MaximumNumberOfUsers" minOccurs="0"/>
                <element ref="msd:MaximumNumberOfDevices" minOccurs="0"/>
```

```
            <element ref="msd:MaximumFrequencyOfUpdateUser" minOccurs="0"/>
            <element ref="msd:MaximumFrequencyOfUpdateDevice" minOccurs="0"/>
            <element ref="msd:UserRevocationList" minOccurs="0"/>
            <element ref="msd:DeviceRevocationList" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 45 — The msdp:CreateDomain element

The message conveys DACredentials, expiration information and optionally user/device related information such as maximum number of them, maximum frequency of update, and the revocation list.

8.5.16 CreateDomainResponse

The msdp:CreateDomainResponse message is sent by the DMD to the Domain Administrator as a response of the msdp:CreateDomain message.

```
<element name="CreateDomainResponse" type="msdp:CreateDomainResponseType"/>
<complexType name="CreateDomainResponseType">
   <complexContent>
      <extension base="msdp:DomainProtocolType">
         <sequence>
            <element ref="msd:DomainID"/>
            <element ref="msd:DomainMembershipCredentials"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 46 — The msdp:CreateDomainResponse element

The message conveys the identifier of the newly created domain, and the DomainMembershipCredentials.

8.5.17 RenewDomain

The msdp:RenewDomain message is sent by a Domain Administrator to the DMD requesting the renewal of a domain when this has expired or the expiration date is approaching.

```
<element name="RenewDomain" type="msdp:RenewDomainType"/>
<complexType name="RenewDomainType">
   <complexContent>
      <extension base="msdp:DomainProtocolType">
         <sequence>
            <element ref="msd:Expiration"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 47 — The msdp:RenewDomain element

This message conveys the expiration information for the renewed domain.

8.5.18 DeleteDomain

The msdp:DeleteDomain message is sent by a Domain Administrator to the DMD requesting to delete a domain.

```
<element name="DeleteDomain" type="msdp:DeleteDomainType"/>
<complexType name="DeleteDomainType">
   <complexContent>
      <extension base="msdp:DomainProtocolType"/>
   </complexContent>
</complexType>
```

Figure 48 — The msdp:DeleteDomain element

8.5.19 UnLicensedSimultaneousUseNotice

The msdp:UnLicensedSimultaneousUseNotice message in Figure 49 is sent by a Media Streaming Player to the DMD for communicating irregularities in the use of domain-bound content.

```
<element name="UnLicensedSimultaneousUseNotice"
type="msdp:UnLicensedSimultaneousUseNoticeType"/>
<complexType name="UnLicensedSimultaneousUseNoticeType">
   <complexContent>
      <extension base="msdp:DomainProtocolType">
         <sequence>
            <element ref="msd:UseData" maxOccurs="unbounded"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 49 — The msdp:UnLicensedSimultaneousUseNotice element

A detailed explanation of the use of the msdp:UnlicensedSimultaneousUseNotice element is given in 8.7.

8.6 Domain Management Protocols specification

8.6.1 Introduction

This section specifies how to employ the messages defined in 8.5 to manage domains. The functionalities of these protocols include:

- setting up a domain described by domain information represented in msd:DomainManageInfo.
- issuing domain membership licenses for devices and users.
- managing devices and users memberships of a domain, i.e. joining, renewing and leaving a domain.
- verifying that domain-bound content has been used within the domain according to license terms.

The figure below shows the flow of the messages necessary to perform the functionalities described above.

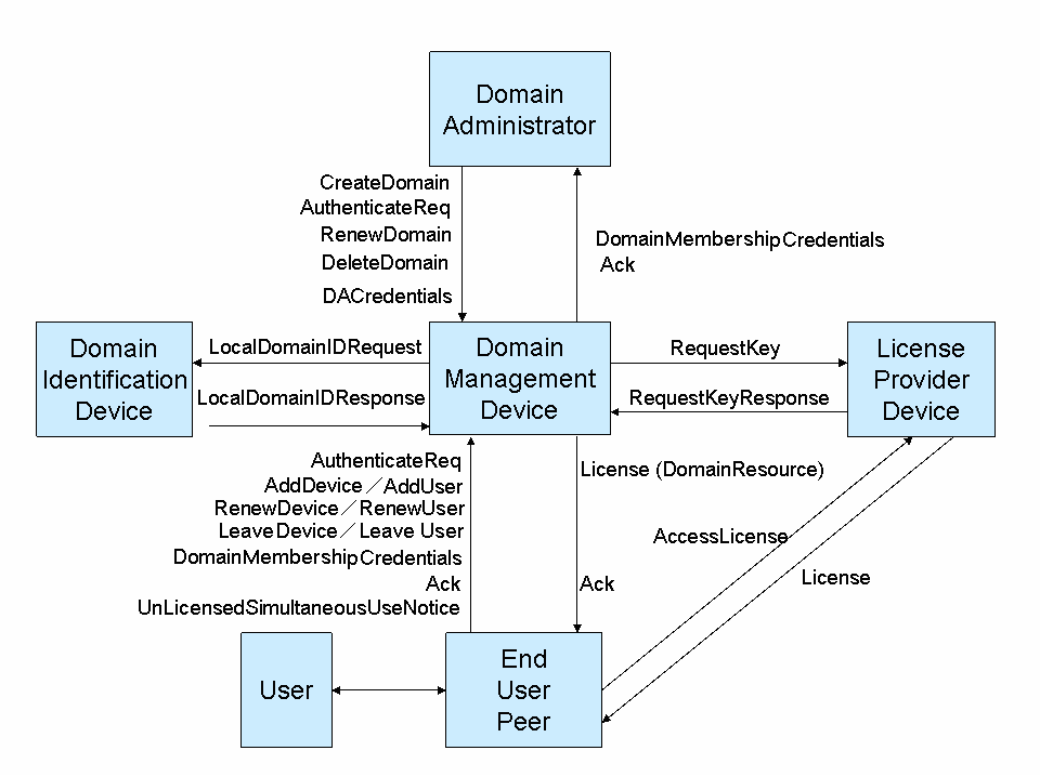

Figure 50 — Schematic Overview of Manage Domain

In summary, the establishment of a Domain is characterised by the following sequence of steps.

Note The detailed protocol specification is given in the following sub-sections.

a) The Domain Administrator requests the creation of a new domain
b) The DMD creates a new domain and registers the domain with a DoID
c) The DoID issues the msd:LocalDomainID and sends it to the DMD
d) The DMD:
    1) generates the msd:DomainID.
    2) creates the domain information, msd:DomainManageInfo.
e) A device or a user joining a domain:
    1) authenticates itself with the DMD, sending the domain membership credentials. (The way account ID and password are communicated to the device or user by the Domain Administrator is not specified).
f) The DMD creates the domain membership license for the device or the user and delivers it to the device or user.

The DMD maintains a list of devices and users for all registered domains being managed. Every time a new device or user joins a domain, the DMD adds the device identifier (msd:DeviceID) or user identifier (msd:UserID) to the corresponding device or user list in msd:DomainManageInfo.

In some circumstances, domain-bound content items may be grouped in Content Groups for being managed within domains. An LPD may employ m1x:isPartOf (See ISO/IEC 21000-5) in a license to manage the simultaneous usage of content in a Content Group by the devices or users. Multiple m1x:isPartOf may appear within a license. Where required, it is possible to limit the number of Content Groups permitted within a domain.

8.6.2 Protocols between the Domain Administrator and the DMD

This subsection specifies the following protocols

1. Create Domain protocol
2. Renew Domain protocol
3. Delete Domain protocol

8.6.2.1 Create Domain Protocol

This protocol is initiated when a Domain Administrator requests the creation of a domain. The protocol for creating a domain is specified below.

a) The Domain Administrator and the DMD mutually authenticate
b) The Domain Administrator generates an msdp:CreateDomain message [Figure 45] containing DACredentials, expiration information and optionally user/device related information such as maximum number of them, maximum frequency of update, and the revocation list and sends it to the DMD
c) The DMD, upon receiving the request from the Domain Administrator, requests DID a new msd:LocalDomainID [Figure 18] by sending a msdp:LocalDomainIDRequest message [Figure 33]
d) If the request can be satisfied, the DID sends an msdp:LocalDomainIDResponse [Figure 34] to the DMD
e) The DMD
    1) creates a number of msd:DomainMembershipCredentials to be given to all Devices and Users allowed to join the domain
    2) generates the domain key, msd:DomainKey
    3) generates and sends the msdp:CreateDomainResponse message [Figure 46] to the Domain Administrator or the msdp:Ack message [Figure 31] to acknowledge an unsuccessful completion of the Protocol.
f) the Domain Administrator:
    1) stores the msd:DomainMembershipCredentials

A Domain Administrator will be given as many pairs of AccessID and AccessPassword as the number of domains he has requested.

8.6.2.2 Renew Domain Protocol

Renewing a domain can be invoked after a domain has expired or when a domain is about to expire. The Domain Administrator requests the DMD to renew the msd:DomainManageInfo.

a) Domain Administrator and DMD mutually Authenticate
b) Domain Administrator issues msdp:AuthenticateReq [Figure 32], identifying the domain to be renewed.
c) DMD returns msdp:Ack [Figure 31].
d) Domain Administrator issues msdp:RenewDomain [Figure 47].
e) DMD
    1) renews msd:DomainManageInfo
    2) (optionally) returns an msdp:Ack [Figure 31] acknowledging the success of the Protocol or otherwise.

8.6.2.3 Delete Domain Protocol

The Domain Administrator requests the deletion of a domain to DMD. The protocol for deleting a domain is shown below.

a) Domain Administrator and DMD mutually Authenticate
b) Domain Administrator issues msdp:AuthenticateReq [Figure 32] , identifying the domain to be deleted.
c) DMD returns msdp:Ack [Figure 31].
d) Domain Administrator issues DeleteDomain [Figure 48]
e) DMD deletes Domain
f) (optionally) DMD returns msdp:Ack [Figure 31] acknowledging the success of the protocol or otherwise.

8.6.3 Protocol between the DMD and the LPD

8.6.3.1 DMD-LPD protocol

This section is an informative part describing how the communication between a DMD and an LPD could be achieved, thus enabling an LPD to issue domain-bound licenses for content. When an LPD issues a license for content bound to a specific domain, it requests the DMD the public key(s) for the domain.

a) DMD and LPD mutually Authenticate
b) LPD issues msdp:RequestKey [Figure 35]
c) DMD delivers the key to the LPD using msdp:RequestKeyResponse [Figure 36]
d) (optionally) LPD replies with msdp:Ack [Figure 31].

Successful execution of this protocol allows the LPD to issue licenses to domain-bound content.

8.6.4 Protocols between the device/User and the DMD

This subsection collects the following Protocols

a) Add Device Protocol
b) Add User Protocol
c) Renew Device Protocol
d) Renew User Protocol
e) Leave Device Protocol
f) Leave User Protocol

8.6.4.1 Add Device Protocol

A device may request the DMD to be added to a domain. If the request can be satisfied, the DMD will add the device unless the number of devices exceeds the maximum number of devices defined in msd:Device part of the msd:DomainManageInfo. The protocol for adding a device is defined below.

a) device and DMD mutually authenticate
b) device issues msdp:AuthenticateReq [Figure 32].
c) DMD returns msdp:Ack [Figure 31].
d) Device issues msdp:AddDevice [Figure 37].
e) DMD adds the device identifier to the list of devices in msd:Device [Figure 26] part of msd:DomainManageInfo.
f) DMD sends a domain membership license to the device contained in the msdp:AddDeviceResponse message [Figure 41]
g) The device (optionally) replies with msdp:Ack [Figure 31].

Successful execution of the Add Device Protocol adds a new DeviceID to the Device ID list of the DMD and adds a new license in the device which from this moment onwards, and until the domain membership does not expire, will be enabled to access domain-bound content.

8.6.4.2 Add User Protocol

A user may request the DMD to be added to a domain. If the request can be satisfied, the DMD will add the user unless the number of users exceeds maximum number of users defined in msd:User part of msd:DomainManageInfo. The protocol for adding a user is defined below.

a) User and DMD mutually authenticate
b) User issues msdp:AuthenticateReq [Figure 32].
c) DMD returns msdp:Ack [Figure 31].
d) User issues msdp:AddUser [Figure 38].
e) DMD adds the user identifier to the user list in msd:User [Figure 24] part of msd:DomainManageInfo.
f) DMD sends a domain membership license to the user contained in the msdp:AddDeviceResponse message [Figure 41]
g) The user (optionally) replies with msdp:Ack [Figure 31].

Successful execution of the Add Device protocol adds a new user to the list of user identifiers of the DMD and adds a new license to the user's secure storage.

8.6.4.3 Renew Device Protocol

The protocol to renew a device is performed when a license granting domain membership to a device expires and the device requests the DMD to renew it. The protocol for renewing the membership of a device to a domain is defined below.

a) Device and DMD mutually authenticate
b) Device issues msdp:AuthenticateReq [Figure 32]
c) DMD returns msdp:Ack [Figure 31]
d) Device issues msdp:RenewDevice [Figure 39]
e) If DMD does not detect unLicensed simultaneous use from the Use Data then DMD:
    1) issues a new domain membership license containing for the device
    2) sends the license to the device contained in a msdp:RenewDeviceResponse message [Figure 41]
f) Else DMD applies policy and sends a msdp:Ack [Figure 3] with attribute "Result" set to "false" to the device
g) The device (optionally) replies with msdp:Ack [Figure 31].

Any Use Data recorded in the device is included in msdp:RenewDevice. If the DMD does not detect un-licensed simultaneous use from the Use Data (see 8.7), the DMD sends a new domain membership license containing for the device with a new expiration period.

8.6.4.4 Renew User Protocol

Renewing a user is invoked when a user domain membership license expires and the user requests the DMD to renew it.

The protocol for renewing a user membership to a domain is defined below.

a) User and DMD mutually authenticate
b) User issues msdp:AuthenticateReq [Figure 32].
c) DMD returns msdp:Ack [Figure 31]
d) User issues msdp:RenewUser [Figure 40]
e) If the user request can be satisfied, DMD issues a new domain membership license for the user and sends it to the user, or applies policy and sends a msdp:Ack [Figure 3] with attribute "Result" set to "false" to the User.
f) The user (optionally) replies with msdp:Ack [Figure 31].

8.6.4.5 Leave Device Protocol

To leave a domain, a device sends a request to the DMD with the purpose of ceasing its membership to a domain. The protocol is performed

1. Device and DMD mutually authenticate
2. Device issues msdp:AuthenticateReq [Figure 32].
3. DMD returns msdp:Ack [Figure 31]
4. Device issues msdp:LeaveDevice [Figure 43]
5. DMD
    a. removes the device identifier from the msd:Device part of msd:DomainManageInfo
    b. replies with msdp:Ack [Figure 31] with the attribute "Result" indicating whether the requested operation was successful or otherwise
6. Device deletes the domain membership license containing in the device.

Successful execution of the Leave Device Protocol erases the domain membership license in the device and the removes the device identifier from the list of devices members of a domain managed by the DMD.

8.6.4.6 Leave User Protocol

A user member of a domain may request DMD to leave a domain. Following this request, the DMD removes the requesting user from the domain.

1. User and DMD mutually authenticate
2. User issues msdp:AuthenticateReq [Figure 32].
3. DMD returns msdp:Ack [Figure 31]
4. User issues msdp:LeaveUser [Figure 44].
5. DMD
    a. removes the user identifier from the list of users part of msd:DomainManageInfo
    b. replies with msdp:Ack [Figure 31] with the attribute "Result" indicating whether the requested operation was successful or otherwise
6. User deletes the domain membership license

Successful execution of the Leave User Protocol erases the domain membership license for the user in the user's secure storage and the user identifier from the list of users members of a domain managed by the DMD.

8.7 Simultaneous Content Usage Detection protocol specification

8.7.1 General

Flexible content licensing models can be implemented through the use of Content Groups, i.e. a set of content items with a common identifier.

A license for a content item or Content Group may be granted to any device in a domain with the condition that the content item belongs to a Content Group. In this case such content item can be used by as many devices in the domain as the number of Content Groups this content item belongs to. In order to verify that this condition is fulfilled, the DMD and the devices belonging to the domain must have at least intermittent connections, so that the DMD can get the Use Data of the devices and verify that no simultaneous use of content has occurred since the last time the DMD and the devices were connected. A trusted clock is clearly also required.

A content item belongs to a Content Group any time it is represented in the correspondent license by means of a resource identifier in m2x:isPartOf element in a License, which is a child element of a m2x:simultaneousAccess condition.

Note Assigning a content item to a Content Group implies that this item cannot be used within the domain at the same time as other items of the same Content Group are being used. This mechanism shall not be confused with the m2x:count condition, which only limits the use of the content item to a number of devices at a time, irrespective of other content items. Setting m2x:count to a specific value is equivalent to a content item being in its own unique group, and therefore having no other items but itself to consider when testing for simultaneous use. The MS MAF at the moment does not specify a mechanism to enforce such condition.

This section specifies a protocol to detect when two devices belonging to the same domain use a content item simultaneously.

8.7.2 Use Data

In order to implement the Simultaneous Usage Detection mechanism, the device must record Use Data on the device i.e. by adding a record for each content use event. Each record consists of a msd:UseData element, and contains the information that the device identified by msd:DeviceID used a content item belonging to m1x:isPartOf in the time frame between “Start Time” and “End Time”. It may include msd:UserID used the content item.

If the device leaves a domain, the device must hold the Use Data of that domain in case it needs to rejoin the same domain. Note that Use Data does not contain the identifier of the content item being used, but rather the Content Group ID to which the content item belongs to.

8.7.3 Merging Use Data between Devices

When two or more devices belonging to the same domain are connected, each device shall merge its Use Data with those of the other devices, in order to increase the possibilities that the DMD discovers un-licensed simultaneous use of content before a device connects to renew its membership to a domain.

Each device adds the Use Data of the other devices to its Use Data. If this operation causes duplicated records in the Use Data, the duplication will be deleted from the Use Data. If two records have the same information except for the value of the Notification Flag element [Figure 29], the record having NotificationFlag with a "FALSE" value will be overwritten by the record with a Notification Flag set to TRUE. Eventually all devices will integrate the Use Data of each other.

When the devices are connected via an unsecured network or server, the Use Data must be exchanged over a secure authenticated channel (SAC).

Note A secure authenticated channel (SAC) is a protocol between two authenticated parties which enables the two to exchange confidential information

8.7.4 Un-Licensed Simultaneous Use

All the devices in a domain that simultaneously used content elements part of a content item belonging to a Content Group are considered as having performed un-licensed Simultaneous Use. This is detected by devices or DMD through inspection of the Use Data.

If the Use Data has any overlapped records in time then the devices indicated by the DeviceID in the overlapped record are considered to have incurred un-Licensed Simultaneous Use. However, if a content item is licensed to more than one m1x:isPartOf, then the use is allocated to the m1x:isPartOf that avoids un-Licensed Simultaneous Use. This is assessed at the time the use data is merged or the totality of Use Data are delivered to the DMD.

The Figure below describes how un-licensed Simultaneous Use is detected. In this figure, device A uses content1 during the period of time shown by the thick line. Similarly device B is shown using content2 during the thick line segment of the lower line. In this case, the usage of content1 on device A and the usage of content2 on device B occur at the same time, shown by the overlapping of their respective thick line segments. Both content1 and content2 belongs to Content Group “CG1” and for this reason only one content item from the Group shall be used at the same time. Device A and Device B makes unlicensed simultaneous usage of content.

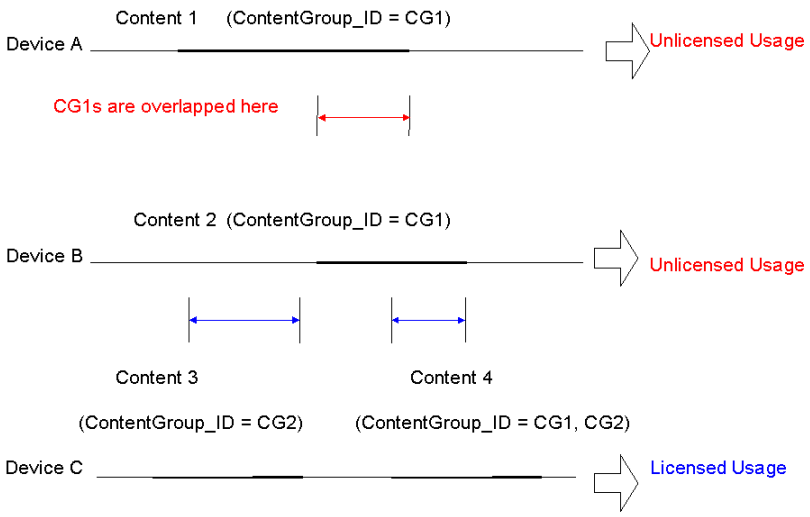

Figure 51 — Un-Licensed Simultaneous Use Schematic

The figure above also shows device C using firstly content3 and later content4 during the time shown by the two thick line segments of the device C timeline. In this case, the usage of content1 on device A and the usage of content3 on device C overlap in time. However, content1 and content3 are given two Content Group ID values so this case does not violate the rule.

The usage of content2 on device B and the usage of content4 on device C also overlaps in time. As content4 is given both “CG1” and “CG2”, “CG2” will be chosen in order to avoid unnecessary detection of Un-Licensed Simultaneous Usage.

8.7.5 Notification to Domain Management Device

Whenever a device connects to the DMD it notifies the DMD if simultaneous use has been detected. This is done by employing the msdp:UnLicensedSimultaneousUseNotice message [Figure 49], containing all msdp:UseData elements where an overlapping was detected by the device. The DMD replies by sending a msbp:Ack [Figure 3] acknowledging the receipt of the message, and applies its own policy.

Once an un-Licensed Simultaneous Use notice is sent to DMD, the Notification Flag element of the record in the Use Data is set to “true” by the device, so that an un-Licensed Simultaneous Use Notice will not be issued again when all the records involved in the simultaneous use have a ‘True’ Notification Flag.

Note The r:validityInterval condition in the license containing the DomainResource for the device or user is set by the DMD to ensure the device or the user will renew its domain membership with an appropriate frequency. According to policy, the Domain Manager may decide to add the device ID to the revocation list.

Annex A
(informative)

Protocol Description Schemas

A.1 Base Protocol Representation schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:mpeg:maf:schema:mediastreaming:baseprotocol:2007"
      xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:msbp="urn:mpeg:maf:schema:mediastreaming:baseprotocol:2007"
      elementFormDefault="qualified" attributeFormDefault="unqualified">
   <complexType name="ProtocolBaseType" abstract="true"/>
   <complexType name="ProtocolType" abstract="true">
      <complexContent>
         <extension base="msbp:ProtocolBaseType">
            <sequence>
               <element name="TransactionID" type="string"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                            Ack                             -->
   <!-- ********************************************************** -->
   <element name="Ack" type="msbp:AckType"/>
   <complexType name="AckType">
      <complexContent>
         <extension base="msbp:ProtocolType">
            <sequence minOccurs="0">
               <element ref="msbp:ProtocolResult"/>
            </sequence>
            <attribute name="Result" type="boolean" use="required"/>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                         ProtocolResult                     -->
   <!-- ********************************************************** -->
   <element name="ProtocolResult" type="msbp:ProtocolResultType"/>
   <complexType name="ProtocolResultType">
      <complexContent>
         <extension base="msbp:ProtocolBaseType">
            <sequence>
               <choice>
                  <element name="ResultCode" type="msbp:ResultCodeType"/>
                  <element name="UserDefinedResult" type="string"/>
               </choice>
               <element name="DisplayString" type="string" minOccurs="0"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <simpleType name="ResultCodeType">
      <annotation>
```

```
            <documentation>
            "00" - RESERVED
            "01" - OK
            "02" - UNKNOWN_MESSAGE
            "03" - TIMEOUT
            "04" - UNABLE_TO_PROCESS
            "05" - UNKNOWN_FAILURE
            "06" - PERMISSION_DENIED
            "07" - BUSY
            </documentation>
         </annotation>
         <restriction base="hexBinary">
            <enumeration value="00"/>
            <enumeration value="01"/>
            <enumeration value="02"/>
            <enumeration value="03"/>
            <enumeration value="04"/>
            <enumeration value="05"/>
            <enumeration value="06"/>
            <enumeration value="07"/>
         </restriction>
      </simpleType>
</schema>
```

A.2 Domain Information Representation schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:mpeg:maf:schema:mediastreaming:domain:2007"
       xmlns="http://www.w3.org/2001/XMLSchema"
       xmlns:msd="urn:mpeg:maf:schema:mediastreaming:domain:2007"
       xmlns:sx="urn:mpeg:mpeg21:2003:01-REL-SX-NS"
       xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
       xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
       xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
       elementFormDefault="qualified" attributeFormDefault="unqualified">
   <complexType name="DomainBaseType" abstract="true"/>
   <complexType name="IDType">
      <sequence>
         <choice>
            <element name="id" type="anyURI"/>
            <element ref="dsig:X509Data" minOccurs="0"/>
         </choice>
      </sequence>
   </complexType>
   <element name="DomainID" type="msd:DomainIDType"/>
   <complexType name="DomainIDType">
      <complexContent>
         <extension base="msd:IDType">
            <sequence>
               <element ref="msd:DomainManagerID"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <element name="DACredentials" type="msd:DomainCredentialType"/>
   <element name="DomainMembershipCredentials" type="msd:DomainCredentialType"/>
   <complexType name="DomainCredentialType">
```

```
      <sequence>
         <element ref="msd:AccessID"/>
         <element ref="msd:AccessPassword"/>
      </sequence>
   </complexType>
   <element name="DomainManageInfo" type="msd:DomainManageInfoType"/>
   <complexType name="DomainManageInfoType">
      <complexContent>
         <extension base="msd:DomainBaseType">
            <sequence>
               <element ref="msd:DomainID"/>
               <element ref="msd:DACredentials" minOccurs="0"/>
               <element ref="msd:DomainMembershipCredentials" minOccurs="0"/>
               <choice minOccurs="0" maxOccurs="2">
                  <element ref="msd:User"/>
                  <element ref="msd:Device"/>
               </choice>
               <element ref="msd:DomainKey"/>
               <element name="Registration" type="dateTime"/>
               <element ref="msd:Expiration"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <element name="User" type="msd:UserType"/>
   <complexType name="UserType">
      <complexContent>
         <extension base="msd:DomainBaseType">
            <sequence>
               <element ref="msd:UserIDList"/>
               <element ref="msd:MaximumNumberOfUsers" minOccurs="0"/>
               <element ref="msd:MaximumFrequencyOfUpdateUser" minOccurs="0"/>
               <element ref="msd:UserRevocationList" minOccurs="0"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <element name="Device" type="msd:DeviceType"/>
   <complexType name="DeviceType">
      <complexContent>
         <extension base="msd:DomainBaseType">
            <sequence>
               <element ref="msd:DeviceIDList"/>
               <element ref="msd:MaximumNumberOfDevices" minOccurs="0"/>
               <element ref="msd:MaximumFrequencyOfUpdateDevice"
minOccurs="0"/>
               <element ref="msd:DeviceRevocationList" minOccurs="0"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <element name="UserRevocationList" type="msd:UserRevocationListType"/>
   <complexType name="UserRevocationListType">
      <sequence minOccurs="0" maxOccurs="unbounded">
         <element ref="msd:UserID"/>
      </sequence>
   </complexType>
   <element name="DeviceIDList" type="msd:DeviceIDListType"/>
   <complexType name="DeviceIDListType">
      <sequence minOccurs="0" maxOccurs="unbounded">
```

```
            <element ref="msd:DeviceID"/>
            <element ref="msd:Expiration"/>
        </sequence>
    </complexType>
    <element name="UserIDList" type="msd:UserIDListType"/>
    <complexType name="UserIDListType">
        <sequence minOccurs="0" maxOccurs="unbounded">
            <element ref="msd:UserID"/>
            <element ref="msd:Expiration"/>
        </sequence>
    </complexType>
    <element name="DeviceRevocationList" type="msd:DeviceRevocationListType"/>
    <complexType name="DeviceRevocationListType">
        <sequence minOccurs="0" maxOccurs="unbounded">
            <element ref="msd:DeviceID"/>
        </sequence>
    </complexType>
    <element name="UseData" type="msd:UseDataType"/>
    <complexType name="UseDataType">
        <sequence>
            <element ref="msd:DomainID"/>
            <element ref="msd:Record" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
    </complexType>
    <element name="Record" type="msd:RecordType"/>
    <complexType name="RecordType">
        <sequence>
            <element ref="msd:DeviceID"/>
            <element name="StartTime" type="dateTime"/>
            <element name="EndTime" type="dateTime"/>
            <element name="NumberOfContentGroups" type="integer"/>
            <element ref="msd:ContentGroupID" minOccurs="0" maxOccurs="unbounded"/>
            <element name="NotificationFlag" type="boolean"/>
        </sequence>
    </complexType>
    <element name="DomainManagerID" type="r:KeyHolder"/>
    <element name="AccessPassword" type="string"/>
    <element name="AccessID" type="string"/>
    <element name="DomainKey" type="xenc:EncryptedKeyType"/>
    <element name="UserID" type="msd:IDType"/>
    <element name="DeviceID" type="msd:IDType"/>
    <element name="LocalDomainID" type="msd:IDType"/>
    <element name="ContentGroupID" type="anyURI"/>
    <element name="MaximumNumberOfDevices" type="unsignedInt"/>
    <element name="MaximumNumberOfUsers" type="unsignedInt"/>
    <element name="MaximumFrequencyOfUpdateDevice" type="duration"/>
    <element name="MaximumFrequencyOfUpdateUser" type="duration"/>
    <element name="Expiration" type="sx:ValidityTimeMetered"/>
</schema>
```

Figure A.1 — The msd namespace

A.3 Domain Protocol Information Representation schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:mpeg:maf:schema:mediastreaming:domainprotocol:2007"
       xmlns="http://www.w3.org/2001/XMLSchema"
       xmlns:msdp="urn:mpeg:maf:schema:mediastreaming:domainprotocol:2007"
```

```
      xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
      xmlns:msd="urn:mpeg:maf:schema:mediastreaming:domain:2007"
      xmlns:msbp="urn:mpeg:maf:schema:mediastreaming:baseprotocol:2007"
      elementFormDefault="qualified" attributeFormDefault="unqualified">
   <!-- ************************************************************** -->
   <!--                      DomainProtocolType                        -->
   <!-- ************************************************************** -->
   <complexType name="DomainProtocolType" abstract="true">
      <complexContent>
         <extension base="msbp:ProtocolType"/>
      </complexContent>
   </complexType>
   <!-- ************************************************************** -->
   <!--                             Ack                                -->
   <!-- ************************************************************** -->
   <element name="Ack" type="msdp:AckType"/>
   <complexType name="AckType">
      <complexContent>
         <extension base="msdp:DomainProtocolType">
            <sequence minOccurs="0">
               <element ref="msbp:ProtocolResult"/>
            </sequence>
            <attribute name="Result" type="boolean" use="required"/>
         </extension>
      </complexContent>
   </complexType>
   <!-- ************************************************************** -->
   <!--                       AuthenticateReq                          -->
   <!-- ************************************************************** -->
   <element name="AuthenticateReq" type="msdp:AuthenticateReqType"/>
   <complexType name="AuthenticateReqType">
      <complexContent>
         <extension base="msdp:DomainProtocolType">
            <sequence>
               <element ref="msd:DomainID" minOccurs="0"/>
               <choice>
                  <element ref="msd:DACredentials"/>
                  <element ref="msd:DomainMembershipCredentials"/>
               </choice>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ************************************************************** -->
   <!--                    LocalDomainIDRequest                        -->
   <!-- ************************************************************** -->
   <element name="LocalDomainIDRequest" type="msdp:RequestLocalDomainIDType"/>
   <complexType name="RequestLocalDomainIDType">
      <complexContent>
         <extension base="msdp:DomainProtocolType"/>
      </complexContent>
   </complexType>
   <!-- ************************************************************** -->
   <!--                   LocalDomainIDResponse                        -->
   <!-- ************************************************************** -->
   <element name="LocalDomainIDResponse" type="msdp:LocalDomainIDResponseType"/>
   <complexType name="LocalDomainIDResponseType">
      <complexContent>
         <extension base="msdp:DomainProtocolType">
            <sequence>
```

```
                <element ref="msd:LocalDomainID"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                        RequestKey                          -->
   <!-- ********************************************************** -->
   <element name="RequestKey" type="msdp:RequestKeyType"/>
   <complexType name="RequestKeyType">
      <complexContent>
         <extension base="msdp:DomainProtocolType">
            <sequence>
               <element ref="msd:DomainID"/>
               <element ref="msd:ContentGroupID" minOccurs="0"
maxOccurs="unbounded"/>
               <choice minOccurs="0" maxOccurs="unbounded">
                  <element ref="msd:DeviceID"/>
                  <element ref="msd:UserID"/>
               </choice>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                    RequestKeyResponse                      -->
   <!-- ********************************************************** -->
   <element name="RequestKeyResponse" type="msdp:RequestKeyResponseType"/>
   <complexType name="RequestKeyResponseType">
      <complexContent>
         <extension base="msdp:DomainProtocolType">
            <sequence>
               <element ref="msd:DomainKey"/>
               <element ref="msd:UserID" minOccurs="0" maxOccurs="unbounded"/>
               <element ref="msd:DeviceID" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                         AddDevice                          -->
   <!-- ********************************************************** -->
   <element name="AddDevice" type="msdp:AddDeviceType"/>
   <complexType name="AddDeviceType">
      <complexContent>
         <extension base="msdp:DomainProtocolType">
            <sequence>
               <element ref="msd:DeviceID"/>
               <element ref="msd:Expiration" minOccurs="0"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                          AddUser                           -->
   <!-- ********************************************************** -->
   <element name="AddUser" type="msdp:AddUserType"/>
   <complexType name="AddUserType">
      <complexContent>
```

```
          <extension base="msdp:DomainProtocolType">
             <sequence>
                <element ref="msd:UserID"/>
                <element ref="msd:Expiration" minOccurs="0"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <!-- ********************************************************************* -->
    <!--                          RenewDevice                                  -->
    <!-- ********************************************************************* -->
    <element name="RenewDevice" type="msdp:RenewDeviceType"/>
    <complexType name="RenewDeviceType">
       <complexContent>
          <extension base="msdp:DomainProtocolType">
             <sequence>
                <element ref="msd:DeviceID"/>
                <element ref="msd:UseData" minOccurs="0"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <!-- ********************************************************************* -->
    <!--                           RenewUser                                   -->
    <!-- ********************************************************************* -->
    <element name="RenewUser" type="msdp:RenewUserType"/>
    <complexType name="RenewUserType">
       <complexContent>
          <extension base="msdp:DomainProtocolType">
             <sequence>
                <element ref="msd:UserID"/>
                <element ref="msd:UseData" minOccurs="0"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <!--
************************************************************************** -->
    <!-- AddDeviceResponse, AddUserResponse, RenewDeviceResponse,
RenewUserResponse -->
    <!--
************************************************************************** -->
    <element name="AddDeviceResponse" type="msdp:LicenseResponseType"/>
    <element name="AddUserResponse" type="msdp:LicenseResponseType"/>
    <element name="RenewDeviceResponse" type="msdp:LicenseResponseType"/>
    <element name="RenewUserResponse" type="msdp:LicenseResponseType"/>
    <complexType name="LicenseResponseType">
       <complexContent>
          <extension base="msdp:DomainProtocolType">
             <sequence>
                <element ref="r:license"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <!-- ********************************************************************* -->
    <!--                          LeaveDevice                                  -->
    <!-- ********************************************************************* -->
    <element name="LeaveDevice" type="msdp:LeaveDeviceType"/>
    <complexType name="LeaveDeviceType">
```

```
      <complexContent>
         <extension base="msdp:DomainProtocolType">
            <sequence>
               <element ref="msd:DeviceID"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************************* -->
   <!--                          LeaveUser                                    -->
   <!-- ********************************************************************* -->
   <element name="LeaveUser" type="msdp:LeaveUserType"/>
   <complexType name="LeaveUserType">
      <complexContent>
         <extension base="msdp:DomainProtocolType">
            <sequence>
               <element ref="msd:UserID"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************************* -->
   <!--                        CreateDomain                                   -->
   <!-- ********************************************************************* -->
   <element name="CreateDomain" type="msdp:CreateDomainType"/>
   <complexType name="CreateDomainType">
      <complexContent>
         <extension base="msdp:DomainProtocolType">
            <sequence>
               <element ref="msd:DACredentials"/>
               <element ref="msd:Expiration"/>
               <element ref="msd:MaximumNumberOfUsers" minOccurs="0"/>
               <element ref="msd:MaximumNumberOfDevices" minOccurs="0"/>
               <element ref="msd:MaximumFrequencyOfUpdateUser" minOccurs="0"/>
               <element ref="msd:MaximumFrequencyOfUpdateDevice"
minOccurs="0"/>
               <element ref="msd:UserRevocationList" minOccurs="0"/>
               <element ref="msd:DeviceRevocationList" minOccurs="0"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************************* -->
   <!--                    CreateDomainResponse                               -->
   <!-- ********************************************************************* -->
   <element name="CreateDomainResponse" type="msdp:CreateDomainResponseType"/>
   <complexType name="CreateDomainResponseType">
      <complexContent>
         <extension base="msdp:DomainProtocolType">
            <sequence>
               <element ref="msd:DomainID"/>
               <element ref="msd:DomainMembershipCredentials"
maxOccurs="unbounded"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************************* -->
   <!--                         RenewDomain                                   -->
   <!-- ********************************************************************* -->
```

```
    <element name="RenewDomain" type="msdp:RenewDomainType"/>
    <complexType name="RenewDomainType">
       <complexContent>
          <extension base="msdp:DomainProtocolType">
             <sequence>
                <element ref="msd:Expiration"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                        DeleteDomain                        -->
    <!-- ********************************************************** -->
    <element name="DeleteDomain" type="msdp:DeleteDomainType"/>
    <complexType name="DeleteDomainType">
       <complexContent>
          <extension base="msdp:DomainProtocolType"/>
       </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--              UnLicensedSimultaneousUseNotice               -->
    <!-- ********************************************************** -->
    <element name="UnLicensedSimultaneousUseNotice"
type="msdp:UnLicensedSimultaneousUseNoticeType"/>
    <complexType name="UnLicensedSimultaneousUseNoticeType">
       <complexContent>
          <extension base="msdp:DomainProtocolType">
             <sequence>
                <element ref="msd:UseData" maxOccurs="unbounded"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
</schema>
```

Figure A.2 — The msdp namespace

A.4 Access Protocols Representation schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:mpeg:maf:schema:mediastreaming:accessprotocol:2007"
       xmlns="http://www.w3.org/2001/XMLSchema"
       xmlns:msap="urn:mpeg:maf:schema:mediastreaming:accessprotocol:2007"
       xmlns:didl-msx="urn:mpeg:maf:schema:mediastreaming:DIDLextensions"
       xmlns:didl="urn:mpeg:mpeg21:2006:07-DIDL-NS"
       xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
       xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
       xmlns:ipmpinfo-
msx="urn:mpeg:maf:Schema:mediastreaming:IPMPINFOextensions:2007"
       xmlns:ipmpinfo="urn:mpeg:mpeg21:2004:01-IPMPINFO-NS"
       xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
       xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
       xmlns:msbp="urn:mpeg:maf:schema:mediastreaming:baseprotocol:2007"
       elementFormDefault="qualified" attributeFormDefault="unqualified">
    <!-- ********************************************************** -->
    <!--                      AccessProtocolType                    -->
    <!-- ********************************************************** -->
    <complexType name="AccessProtocolType" abstract="true">
       <complexContent>
```

```
            <extension base="msbp:ProtocolType"/>
         </complexContent>
      </complexType>
      <!-- ********************************************************** -->
      <!--                              Ack                           -->
      <!-- ********************************************************** -->
      <element name="Ack" type="msap:AckType"/>
      <complexType name="AckType">
         <complexContent>
            <extension base="msap:AccessProtocolType">
               <sequence minOccurs="0">
                  <element ref="msbp:ProtocolResult"/>
               </sequence>
               <attribute name="Result" type="boolean" use="required"/>
            </extension>
         </complexContent>
      </complexType>
      <!-- ********************************************************** -->
      <!--                    ContentIdentifierType                   -->
      <!-- ********************************************************** -->
      <complexType name="ContentIdentifierType">
         <complexContent>
            <extension base="msbp:ProtocolBaseType">
               <sequence>
                  <element name="ContentItemIdentifier" type="anyURI"/>
                  <element name="ContentElementIdentifier" type="anyURI"
minOccurs="0"/>
               </sequence>
            </extension>
         </complexContent>
      </complexType>
      <!-- ********************************************************** -->
      <!--                        RequestContent                      -->
      <!-- ********************************************************** -->
      <element name="RequestContent" type="msap:RequestContentType"/>
      <complexType name="RequestContentType">
         <complexContent>
            <extension base="msap:AccessProtocolType">
               <sequence>
                  <element name="ContentIdentifier"
type="msap:ContentIdentifierType"/>
                  <element name="MimeType" type="string" minOccurs="0"/>
                  <element ref="r:license" minOccurs="0"/>
                  <element name="UsageEnvironmentDescription"
type="dia:UsageEnvironmentType" minOccurs="0"/>
                  <element ref="dsig:Signature" minOccurs="0"/>
               </sequence>
            </extension>
         </complexContent>
      </complexType>
      <!-- ********************************************************** -->
      <!--                    RequestContentResponse                  -->
      <!-- ********************************************************** -->
      <element name="RequestContentResponse"
type="msap:RequestContentResponseType"/>
      <complexType name="RequestContentResponseType">
         <complexContent>
            <extension base="msap:AccessProtocolType">
               <sequence>
                  <element name="DI" type="didl:DIDLType" minOccurs="0"/>
```

```
                <element name="ContentURL" type="msap:ContentURLType"
minOccurs="0" maxOccurs="unbounded"/>
                <element ref="dsig:Signature" minOccurs="0"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                        ContentURLType                      -->
    <!-- ********************************************************** -->
    <complexType name="ContentURLType">
       <complexContent>
          <extension base="msbp:ProtocolBaseType">
             <sequence>
                <element name="MimeType" type="string"/>
                <element name="URL" type="anyURI"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                        RequestLicense                      -->
    <!-- ********************************************************** -->
    <element name="RequestLicense" type="msap:RequestLicenseType"/>
    <complexType name="RequestLicenseType">
       <complexContent>
          <extension base="msap:AccessProtocolType">
             <sequence>
                <choice>
                   <element name="ContentIdentifier"
type="msap:ContentIdentifierType"/>
                   <element name="LicenseID" type="anyURI"/>
                </choice>
                <element ref="r:license" minOccurs="0"/>
                <element ref="dsig:Signature" minOccurs="0"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                    RequestLicenseResponse                  -->
    <!-- ********************************************************** -->
    <element name="RequestLicenseResponse"
type="msap:RequestLicenseResponseType"/>
    <complexType name="RequestLicenseResponseType">
       <complexContent>
          <extension base="msap:AccessProtocolType">
             <sequence>
                <element ref="r:license"/>
                <element ref="dsig:Signature" minOccurs="0"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                     RequestIPMPToolBody                    -->
    <!-- ********************************************************** -->
    <element name="RequestIPMPToolBody" type="msap:RequestIPMPToolBodyType"/>
    <complexType name="RequestIPMPToolBodyType">
       <complexContent>
```

```
          <extension base="msap:AccessProtocolType">
             <sequence>
                <element ref="ipmpinfo:IPMPToolID"/>
                <element ref="ipmpinfo-msx:DeviceInformation"/>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
    <!-- ************************************************************ -->
    <!--                 RequestIPMPToolBodyResponse                  -->
    <!-- ************************************************************ -->
    <element name="RequestIPMPToolBodyResponse"
type="msap:RequestIPMPToolBodyResponseType"/>
    <complexType name="RequestIPMPToolBodyResponseType">
       <complexContent>
          <extension base="msap:AccessProtocolType">
             <sequence>
                <choice maxOccurs="unbounded">
                   <element ref="ipmpinfo-msx:ToolBody"/>
                   <element name="ToolURL" type="anyURI"/>
                </choice>
             </sequence>
          </extension>
       </complexContent>
    </complexType>
</schema>
```

Figure A.3 — The msap namespace

Bibliography

[1] ISO/IEC 21000-2:2005, *Information technology — Multimedia framework (MPEG-21) — Part 2: Digital Item Declaration*

[2] ISO/IEC 21000-3:2003, *Information technology — Multimedia framework (MPEG-21) — Part 3: Digital Item Identification*

[3] ISO/IEC 21000-4:2006, *Information technology — Multimedia framework (MPEG-21) — Part 4: Intellectual Property Management and Protection Components*

[4] ISO/IEC 21000-5, *Information technology — Multimedia framework (MPEG-21) — Part 5: Rights Expression Language — Amendment 2: DAC (Dissemination & Capture) profile*

[5] ISO/IEC 21000-7, *Information technology — Multimedia framework (MPEG-21) — Part 7: Digital Item Adaptation*

[6] ISO/IEC 23001-3, *Information technology — MPEG systems technologies — Part 3: XML IPMP messages*

[7] XML Encryption Syntax and Processing, W3C Recommendation 10 December 2002. http://www.w3.org/TR/xmlenc-core/

[8] XML Signature Syntax and Processing, W3C Recommendation 12 February 2002. http://www.w3.org/TR/xmldsig-core/

[9] XML Schema Part 1: Structures, Second Edition, W3C Recommendation 28 October 2004. http://www.w3.org/2001/XMLSchema

[10] XML Path Language (XPath) 2.0, W3C Recommendation, 23 January 2007. http://www.w3.org/TR/xpath20/

[11] XML Encryption Syntax and Processing, W3C Recommendation 10 December 2002. http://www.w3.org/TR/xmlenc-core/

[12] XML Signature Syntax and Processing, W3C Recommendation 12 February 2002. http://www.w3.org/TR/xmldsig-core/

[13] IETF, Network Working Group, Request for Comments: 2459, Internet X.509 Public Key Infrastructure Certificate and CRL Profile. http://www.ietf.org/rfc/rfc2459.txt

ICS 35.040
Price based on 47 pages