INTERNATIONAL STANDARD

ISO/IEC 23006-4

First edition
2010-12-01

Information technology — MPEG extensible middleware (MXM) —

Part 4: MXM protocols

Technologies de l'information — Intergiciel MPEG extensible (MXM) —

Partie 4: Protocoles MXM

Reference number
ISO/IEC 23006-4:2010(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 23006-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This first edition of ISO/IEC 23006-4 cancels and replaces ISO/IEC 29116-1:2008.

ISO/IEC 23006 consists of the following parts, under the general title *Information technology — MPEG extensible middleware (MXM)*:

— *Part 1: MXM architecture and technologies*

— *Part 2: MXM API*

— *Part 3: MXM conformance and reference software*

— *Part 4: MXM protocols*

Introduction

ISO/IEC 23006 is a suite of standards that has been developed for the purpose of enabling the easy design and implementation of media-handling value chains whose devices interoperate because they are all based on the same set of technologies accessible from the MXM middleware.

This will enable the development of a global market of

— MXM applications that can run on MXM devices thanks to the existence of a standard MXM application API,

— MXM devices executing MXM applications thanks to the existence of a standard MXM architecture,

— MXM engines thanks to the existence of standard MXM architecture and standard APIs, and

— innovative business models because of the ease to design and to implement media-handling value chains whose devices interoperate because they are all based on the same set of technologies, especially WG11 technologies.

Information technology — MPEG extensible middleware (MXM) —

Part 4:
MXM protocols

1 Scope

This part of ISO/IEC 23006 specifies a set of protocols enabling distributed applications to exchange information related to content items and parts thereof, including rights and protection information.

This part of ISO/IEC 23006 specifies five categories of protocols: the content protocols, the license protocols, the IPMP tool protocols, the domain protocols and the event report protocols. The first category includes protocols to access, authenticate, identify and store a content item or parts thereof. The second includes protocols to access and store a license from/to a remote service. The third category includes protocols to access an IPMP tool (a module performing protection operations such as decryption, watermarking, key management, etc.) from a remote service while a fourth category includes the protocols allowing a number of devices to create, join, administer, etc. a group of users and devices where the participants share common properties. The fifth category, event report, comprises the protocol to store an event report and the protocol to register an event report request.

In security-aware environments, the security at the communication level is assumed to be handled by traditional underlying security protocols, e.g. the SSLv3 or TLSv1 protocols. This part of ISO/IEC 23006 does not apply to this level. Figure 1 shows the scope of the MXM protocols.

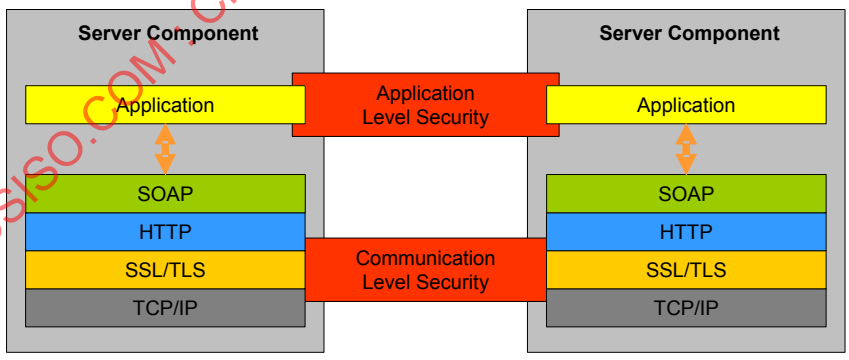

Figure 1 — Scope of MXM protocols

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 23006-1, *Information technology — MPEG extensible middleware (MXM) — Part 1: MXM architecture and technologies*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 23006-1 apply.

4 Abbreviated terms

DID	Digital Item Declaration
DIDL	Digital Item Declaration Language
DII	Digital Item Identification
DMD	Domain Management Device
DoID	Domain Identification Device
IPMP	Intellectual Property Management and Protection
URI	Uniform Resource Identifier

5 Namespace conventions

Throughout this part of ISO/IEC 23006, 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 23006, consistent namespace prefixes are used. Table 1 gives these prefixes and the corresponding namespace.

Table 1 — Namespaces and prefixes

Prefix	Corresponding namespace
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
dsig	http://www.w3.org/2000/09/xmldsig#
ipmpdidl	urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS
ipmpmsg	urn:mpeg:mpeg21:2006:07-IPMPMESSAGES-NS
ipmpinfo	urn:mpeg:mpeg21:2004:01-IPMPINFO-NS
m1x	urn:mpeg:mpeg21:2005:01-REL-M1X-NS
mxmacp	urn:mpeg:mpeg-m:schema:accesscontentprotocol:2009
mxmaitp	urn:mpeg:mpeg-m:schema:accessipmptoolprotocol:2009
mxmalp	urn:mpeg:mpeg-m:schema:accesslicenseprotocol:2009
mxmaucp	urn:mpeg:mpeg-m:schema:authenticatecontentprotocol:2009
mxmbp	urn:mpeg:mpeg-m:schema:baseprotocol:2009
mxmd	urn:mpeg:mpeg-m:schema:domain:2009
mxmdp	urn:mpeg:mpeg-m:schema:domainprotocol:2009

mxmicp	urn:mpeg:mpeg-m:schema:identifycontentprotocol:2009
mxmrlp	urn:mpeg:mpeg-m:schema:revokelicenseprotocol:2009
mxmscp	urn:mpeg:mpeg-m:schema:storecontentprotocol:2009
mxmslp	urn:mpeg:mpeg-m:schema:storelicenseprotocol:2009
r	urn:mpeg:mpeg21:2003:01-REL-R-NS
sx	urn:mpeg:mpeg21:2003:01-REL-SX-NS
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance

6 System overview

ISO/IEC 23006-1 specifies the format of the data exchanged between distributed applications part of media-handling value chains as shown in the figure below.

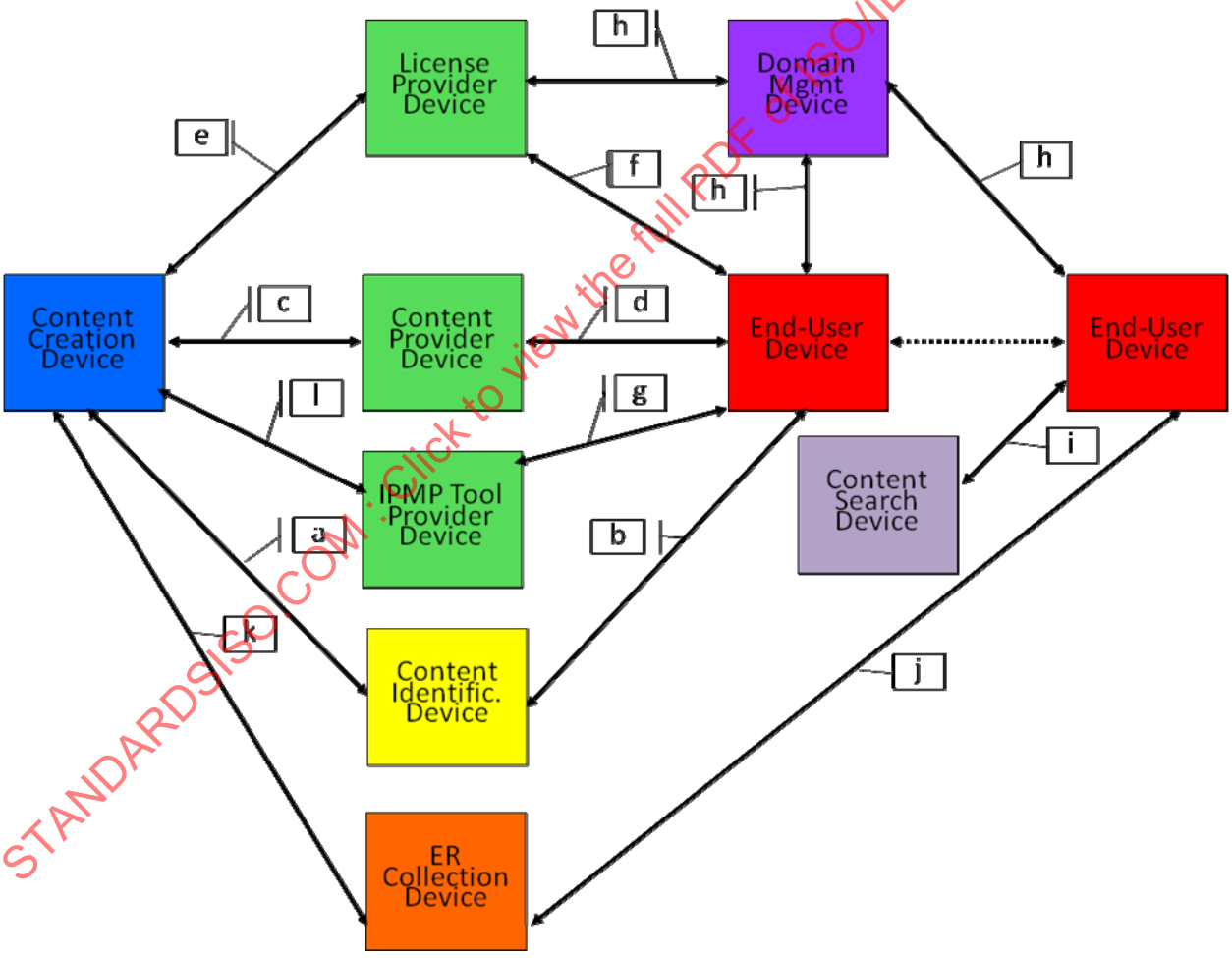

Figure 2 — Typical devices in a media-handling value chain

The devices in Figure 1 are defined as following:

— Content Creation Device, a device capable of creating content items possibly including audio-visual resources, metadata, rights information, etc.

— Content Provider Device, a device capable of storing content and in turn providing it to other devices (e.g. via streaming, downloading, etc.).

— License Provider Device, a device capable of being configured to issue licenses to other devices upon request.

— IPMP Tool Provider Device, a device capable of interacting with other devices to provide IPMP Tools.

— Content Identification Device, a device capable of providing identifiers to new content items and parts thereof, and allowing applications to verify the authenticity of the identified objects.

— Domain Management Device, a device capable of managing various functions needed for a proper functioning of a domain.

— End-User Device, a device capable of accessing content, licenses and IPMP Tools from other devices, authenticating content and becoming part of a domain of devices.

— Event Report Collecting Device, a device capable of processing ER-R and issue ER.

— Content Search Device, a device providing other devices with responses to queries.

The Protocols specified by this International Standard are identified in Figure 1 with a number representing the following:

Table 2 — List of MXM Protocols

#	Protocol name	Purpose
a	Identify Content Protocol	to identify ontent items and elements thereof, as specified in 8.1
b	Authenticate Content Protocol	to authenticate content items and elements thereof, as specified in 8.2
c	Store Content Protocol	to store content items and elements thereof, as specified in 8.3
d	Access Content Protocol	to obtain content items and elements thereof, as specified in 8.4
e	Store License Protocol	to configure a license service to issue licenses, as specified in 9.1
f	Access License Protocol	to obtain licenses granting rights over content items and elements thereof, as specified in 9.3
g	Access IPMP Tool Protocol	to obtain IPMP Tools necessary to access protected content, as specified in 10.1
h	Manage Domain Protocol	to create, join, administer, etc. a group of users and devices, as specified in 11
i	Content Search Protocol	to perform searches for content items having specific characteristics, as specified in ISO/IEC 15938-12
j	Store Event Report Protocol	to request the ECD to store an Event Report, as specified in 12.2.2
k	Register Event Report Request	to request the ECD to register a class of Event Reports triggered by a certain Event Report Request, as specified in 12.2.1
l	Access IPMP Tool List Protocol	to request the list of IPMP Tool Bodies available on the TPD, as specified in 10.2

The Protocols listed above are divided in five categories: Content Protocols, comprising of a), b), c), d) and i, License Protocols comprising of e) and f), IPMP Tool Protocols comprising of g) and l), Manage Domain Protocols comprising of h), “and “Event Report” comprising of j), k)

The messages exchanged between two devices are based on a transactional protocol that is supported over an existing network protocol (e.g. TCP/IP or HTTP in the case of Internet/WWW access).

7 Base Protocol

7.1 Introduction

This Clause specifies the base information commonly used in both the MXM protocols. The namespace *mxmbp* defines the elements on which the access protocols and the domain protocols are based.

7.2 Protocol data format

7.2.1 ProtocolBaseType

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

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

Figure 3 — The mxmbp:ProtocolBaseType complex type

7.2.2 ProtocolType

The abstract mxmbp:ProtocolType complex type, defined in the figure below, extends the mxmbp:ProtocolBaseType for conveying the mxmbp: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="mxmbp:ProtocolBaseType">
         <sequence>
            <element name="TransactionID" type="string"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 4 — The mxmbp:ProtocolType complex type

7.2.3 Ack

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

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

Figure 5 — The mxmbp:Ack element

7.2.4 ProtocolResult

The ProtocolResult element may convey a result code indicating the result of a requested operation, and a string to display to the user indicating more informazion on the result of the operation.

```
<element name="ProtocolResult" type="mxmbp:ProtocolResultType"/>
<complexType name="ProtocolResultType">
   <complexContent>
      <extension base="mxmbp:ProtocolBaseType">
         <sequence>
            <element name="ResultCode" type="mxmbp:ResultCodeType"
minOccurs="0"/>
            <element name="DisplayString" type="string" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 6 — The mxmbp:ProtocolResult element

The ResultCode can be either one of those listed in Table 2 or it can be a new value extending the ExtendableResultCodeType simpleType.

```
<complexType name="ResultCodeType">
   <complexContent>
      <extension base="mxmbp:ProtocolBaseType">
         <sequence>
            <choice>
               <element name="BasicResultCode"
type="mxmbp:BasicResultCodeType"/>
               <element name="ExtendableResultCode"
type="mxmbp:ExtendableResultCodeType"/>
            </choice>
         </sequence>
      </extension>
   </complexContent>
</complexType>
<simpleType name="BasicResultCodeType">
   <restriction base="string">
      <enumeration value="OK"/>
      <enumeration value="PERMISSION_DENIED"/>
      <enumeration value="TIMEOUT"/>
      <enumeration value="BUSY"/>
      <enumeration value="MALFORMED_REQUEST"/>
      <enumeration value="UNABLE_TO_PROCESS"/>
      <enumeration value="OPERATION_NOT_SUPPORTED"/>
      <enumeration value="UNKNOWN_MESSAGE"/>
```

```
        <enumeration value="UNKNOWN_ERROR"/>
    </restriction>
</simpleType>
<simpleType name="ExtendableResultCodeType">
    <restriction base="string"/>
</simpleType>
```

Figure 7 — The mxmbp:ResultCodeType complexType

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

Table 3 — List of Result Codes defined in the Base Protocol schema

Result Code	Semantics
OK	The requested operation was carried out successfully
PERMISSION_DENIED	The sender is not allowed to carry out the requested operation
TIMEOUT	A timeout occurred while carrying out the requested operation
BUSY	The requested operation cannot be performed because the addressee is busy
MALFORMED_REQUEST	The request message is malformed or incomplete
UNABLE_TO_PROCESS	The requested operation is supported by the addressee, however the addressee is not able to process the request for an unknown reason
OPERATION_NOT_SUPPORTED	The requested operation is not supported by the addressee
UNKNOWN_MESSAGE	The sent message was not recognised by the addressee
UNKNOWN_ERROR	An unknown error occurred

7.2.5 ContentIdentifier

The ContentIdentifier element specified in the figure below conveys the identifier of a content item and optionally the identifier of a content element part of the content item. In the case the mxmbp:ContentElementIdentifier element is specified, only the specific content element is requested, and not the whole content item.

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

Figure 8 — The mxmbp:ContentIdentifier element

8 Content Protocols

8.1 Identify Content Protocol

The Identify Content Protocol is employed to register a Digital Item and its main identifiable elements, namely resources, licences and IPMP Tools.

Two versions of the protocols are provided. In the first, the Content Item itself (i.e. the DI) is sent to the Content Identification Device (CID) from the Content Creation Device (CCD). In the second only the hash is communicated.

8.1.1 Protocol specification

8.1.1.1 Identify Content with transfer of DI

This Protocol specifies how to obtain an identifier for a Content Item from a CID. This protocol requires the CCD to send the complete DI to the CID. This Protocol is as follows:

a) CCD and CID mutually Authenticate

b) The CCD sends to the CID an mxmicp:IdentifyContentRequest [8.1.2.3];

c) The CID

   1) Assigns a new Content Identifier to the received DCI

   2) Adds the Identifier to received DCI

   3) Digitally Signs or otherwise Hashes the received DCI

   4) Stores the Content ID and the generated Hash value in the CID database

   5) Returns the modified DCI to requesting party by including it in an mxmicp:IdentifyContentResponse message [8.1.2.4].

8.1.1.2 Identify Content with transfer of Signature/Hash

This Protocol specifies how to obtain an identifier for a Content Item from a CID. This protocol requires the CCD to send only the Hash of the Identified DCI to the CID. This Protocol is as follows:

a) CID and CCD mutually Authenticate

b) CCD request a Content Identifier from the CID by sending a mxmicp:RequestContentIdentifier [8.1.2.5]

c) If the request from the CCD can be satisfied, the CID

   1) generates the requested Identifier

   2) generates an mxmicp:RequestIdentifierResponse message containing the Identifier [8.1.2.7]

d) The CCD

   1) Adds the received Identifier to the DI to be Identified

   2) Computes the hash of the DCI with the Identifier included

   3) Sends a mxmicp:RegisterIdentifier message to the CID [8.1.2.8]

e) The CID

1) Stores the Identifier together with the Hash in the CID database for future reference.

2) Replies with an mxmicp:Ack message to the CCD [8.1.2.2]

8.1.1.3 Protocol to Identify Content Elements

This Protocol specifies how to obtain an identifier for a Content Element (e.g. a media Resource) from a Content Identification Device. This protocol requires the CCD to send the hash value of the Content Element to be Identified. This Protocol is carried out in the following steps:

a) The CCD:

1) generates the Hash value or the digital Signature of the Content Element to be Identified and inserts it in a mxmicp:RequestContentElementIdentifier message [8.1.2.6]

2) sends the message to the CID

b) The CID, if the request from the CCD can be satisfied

1) generates an Identifier

2) stores the Identifier together with the Hash value/digital Signature in the CID database

3) returns an mxmicp:RequestIdentifierResponse message to the CCD [8.1.2.7].

c) The CCD inserts the identifier obtained from the CID into the Digital Item for

8.1.2 Protocol data format

This Subclause specifies the payload of the messages employed by a Device (e.g. a Content Creation Device) to obtain a Content Identifier for a new Content Item from a Content Identification Device (CID, which is likely run by a Content Identification Agency).

8.1.2.1 IdentifyContentProtocolType

The base type for all Content Identifier Protocol messages is defined in the figure below.

```
<complexType name="IdentifyContentProtocolType" abstract="true">
   <complexContent>
      <extension base="mxmbp:ProtocolType"/>
   </complexContent>
</complexType>
```

Figure 9 — The mxmicp:IdentifyContentProtocolType complex type

8.1.2.2 Ack

The mxmicp:Ack message, defined in the figure below, is used to acknowledge the success of an operation or to convey an error message in case of failure.

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

Figure 10 — The mxmicp:Ack element

The mxmicp:Ack message extends the mxmbp:ProtocolResult message by adding the "Result" attribute specifying whether the acknowledged operation was successful or otherwise.

8.1.2.3 IdentifyContentRequest

The mxmicp:IdentifyContentRequest message is specified in the figure below:

```
<element name="IdentifyContentRequest" type="mxmicp:IdentifyContentRequestType"/>
<complexType name="IdentifyContentRequestType">
   <complexContent>
      <extension base="mxmicp:IdentifyContentProtocolType">
         <sequence>
            <element name="DCI" type="didl:DIDLType"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 11 — The mxmicp:IdentifyContentRequest element

This message shall include the Digital Item representing the new content item without any Content ID specified in the dii:Identifier, as this will be generated and inserted by the CID. This message may be digitally signed; in this case the value of the signature shall be inserted in the dsig:Signature element.

The CID will return an IdentifyContentResponse message in response to the CCD.

8.1.2.4 IdentifyContentResponse

The mxmicp:IdentifyContentResponse message is specified in the figure below:

```
<element name="IdentifyContentResponse"
type="mxmicp:IdentifyContentResponseType"/>
<complexType name="IdentifyContentResponseType">
   <complexContent>
      <extension base="mxmicp:IdentifyContentProtocolType">
         <sequence>
            <element name="DCI" type="didl:DIDLType"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 12 — The mxmicp:IdentifyContentResponse element

The mxmicp:IdentifyContentResponse message contains the Digital Item now including the Content ID. The DI conveyed in the response message may have been digitally signed or hashed by the Content Identification Device. Moreover, an optional digital signature can be applied to the whole mxmicp:IdentifyContentResponse message.

8.1.2.5 RequestContentIdentifier

This message is an alternative means by which a Content Creation Devices may request a new Content ID to a Content Identification Device. The mxmicp:RequestContentIdentifier message is specified in the figure below:

```
<element name="RequestContentIdentifier"
type="mxmicp:RequestContentIdentifierType"/>
<complexType name="RequestContentIdentifierType">
    <complexContent>
        <extension base="mxmicp:IdentifyContentProtocolType"/>
    </complexContent>
</complexType>
```

Figure 13 — The mxmicp:RequestContentIdentifier element

8.1.2.6 RequestContentElementIdentifier

A Content Creation Device may request an identifier for a new content element (e.g. an audio-visual resource) to a Content Identification Device by employing the mxmicp:RequestContentElementIdentifier message.

```
<element name="RequestContentElementIdentifier"
type="mxmicp:RequestContentElementIdentifierType"/>
<complexType name="RequestContentElementIdentifierType">
    <complexContent>
        <extension base="mxmicp:IdentifyContentProtocolType">
            <sequence>
                <element name="ContentElementSignature" type="dsig:SignatureType"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 14 — The mxmicp:RequestContentElementIdentifier element

The RequestContentElementIdentifier shall convey the digital signature or hash value of the content element for which an identifier is requested to the Content Identification Device.

8.1.2.7 RequestIdentifierResponse

An mxmicp:RequestIdentifierResponse message is sent to a CCD in response to an mxmicp:RequestContentIdentifier or mxmicp:RequestContentElementIdentifier message.

```
<element name="RequestIdentifierResponse"
type="mxmicp:RequestIdentifierResponseType"/>
<complexType name="RequestIdentifierResponseType">
    <complexContent>
        <extension base="mxmicp:IdentifyContentProtocolType">
            <sequence>
                <element ref="dii:Identifier"/>
                    <element ref="dsig:SignedInfo" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 15 — The mxmicp:RequestIdentifierResponse element

The message conveys the requested Content or Content Element identifier.

8.1.2.8 RegisterIdentifier

The mxmicp:RegisterIdentifier message is sent in response to an mxmicp:RequestIdentifierResponse to convey the digital signature or hash value of the identified DI to the Content Identification Device together with the identifier received in the RequestIdentifierResponse message previously received.

```
<element name="RegisterIdentifier" type="mxmicp:RegisterIdentifierType"/>
<complexType name="RegisterIdentifierType">
   <complexContent>
      <extension base="mxmicp:IdentifyContentProtocolType">
         <sequence>
            <element ref="dii:Identifier"/>
            <element name="DCISignature" type="dsig:SignatureType"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 16 — The mxmicp:RegisterIdentifier element

8.2 Authenticate Content Protocol

8.2.1 Protocol specification

The Protocol to Authenticate Content is employed by any entity wishing to verify the authenticity of a content item or elements thereof. In order to authenticate a content item or content element, the following steps shall be performed:

a) The requesting Device (e.g. the End User Device):

   1) Generate an mxmaucp:AuthenticateContentRequest message [8.2.2.3] containing

      i) either:

         I) the Content ID of the content item to be authenticated

         II) (optionally) the DISignature element containing the hash value of the DI

      ii) or

         I) the full DI representing the content item to be authenticated

   2) Mutually Authenticate with CID

   3) Sends the mxmaucp:AuthenticateContentRequest to the CID

b) The CID:

   1) Parses the received mxmaucp:AuthenticateContentRequest message and if the request can be satisfied generates an mxmaucp:AuthenticateResponse message [8.2.2.5] containing:

      i) either:

         I) the DISignature for the content item (if this exists in the CID database), if the mxmaucp:AuthenticateContentRequest contained the ContentID field. The DISignature may

contain either the digital signature of the DI, or its hash value only, and the signature/hash calculation an verification is left to the requesting device.

ii) or:

I) the AuthenticationResult boolean value, if the mxmaucp:AuthenticateContentRequest contained the whole DI in the didl:DIDL field. In this case the signature/hash calculation and verification is done by the CID

iii) or:

I) An error code specifying that the requested service could not be performed.

8.2.2 Protocol data format

This Subclause specifies the payload of the messages employed by a Device (e.g. an End-User Device) to authenticate a Content Item with a Content Identification Device (CID, which is likely run by a Content Identification Agency).

8.2.2.1 AuthenticateContentProtocolType

The mxmaucp:AuthenticateContentProtocolType complex type, defined in the figure below, extends the mxmbp:ProtocolType.

```
<complexType name="AuthenticateContentProtocolType" abstract="true">
   <complexContent>
      <extension base="mxmbp:ProtocolType"/>
   </complexContent>
</complexType>
```

Figure 17 — The mxmaucp:AuthenticateContentProtocolType complex type

8.2.2.2 Ack

The mxmaucp:Ack message, defined in the figure below, is used to acknowledge the success of an operation or to convey an error message in case of failure.

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

Figure 18 — The mxmaucp:Ack element

The mxmaucp:Ack message extends the mxmbp:ProtocolResult message by adding the “Result” attribute specifying whether the acknowledged operation was successful or otherwise.

8.2.2.3 AuthenticateContentRequest

Any device can authenticate a content item at any time. For this purpose, the following message is defined:

```
<element name="AuthenticateContentRequest"
type="mxmaucp:AuthenticateContentRequestType"/>
<complexType name="AuthenticateContentRequestType">
    <complexContent>
        <extension base="mxmaucp:AuthenticateContentProtocolType">
            <sequence>
                <choice>
                    <element name="DCIInfo" type="mxmaucp:InfoType"/>
                    <element name="DCI" type="didl:DIDLType"/>
                </choice>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="InfoType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <element name="ID" type="anyURI"/>
                <element name="Signature" type="dsig:SignatureType" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 19 — The mxmaucp:AuthenticateContentRequest element

The mxmaucp:AuthenticateContentRequest message conveys a choice between DI information (the Content ID and an optional DI Signature) and the whole DI (conveyed in the didl:DIDL element). The mxmaucp:AuthenticateContentRequest message can be digitally signed; in this case the signature shall be conveyed in the dsig:Signature element.

8.2.2.4 AuthenticateContentElementRequest

Any Device can authenticate a content element (e.g. an audio-visual resource) at any time. For this purpose, the following message is defined:

```
<element name="AuthenticateContentElementRequest"
type="mxmaucp:AuthenticateContentElementRequestType"/>
<complexType name="AuthenticateContentElementRequestType">
    <complexContent>
        <extension base="mxmaucp:AuthenticateContentProtocolType">
            <sequence>
                <element name="ContentElementInfo" type="mxmaucp:InfoType"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 20 — The mxmaucp:AuthenticateContentElementRequest element

The mxmaucp:AuthenticateContentElementRequest message conveys the content element ID and the content element signature.) and the whole DI (conveyed in the didl:DIDL element). The

mxmaucp:AuthenticateContentElementRequest message can be digitally signed; in this case the Signature shall be conveyed in the dsig:Signature element.

8.2.2.5 AuthenticateResponse

Upon receiving either an Authenticate Content or Content Element Request message, a CID shall reply by means of the following message

```
<element name="AuthenticateResponse" type="mxmaucp:AuthenticateResponseType"/>
<complexType name="AuthenticateResponseType">
    <complexContent>
        <extension base="mxmaucp:AuthenticateContentProtocolType">
            <sequence>
                <choice>
                    <element name="ContentSignature" type="dsig:SignatureType"/>
                    <element name="AuthenticationResult" type="boolean"/>
                    <element name="ErrorCode" type="mxmaucp:ErrorCodeType"/>
                </choice>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<simpleType name="ErrorCodeType">
    <restriction base="string">
        <enumeration value="HASH_MISSING"/>
        <enumeration value="HASH_CORRUPTED"/>
    </restriction>
</simpleType>
```

Figure 21 — The mxmaucp:AuthenticateResponse element

The mxmaucp:AuthenticateResponse message conveys a choice between

— ContentSignature: the signature of either the content item or the content element (depending on the request made) in the case the task of verifying the signature or the hash is made by the requesting device

— AuthenticationResult: the result of the authentication

— ErrorCode: in case the AuthenticationResult element is set to "true", this element will convey further information on the reason of failure, which can be either "HASH_MISSING" or "HASH_CORRUPTED".

8.3 Store Content Protocol

8.3.1 Protocol specification

The Protocol to Store Content is employed by a Content Creation Device (CCD) to store a content item on a Content Provider Device (CPD). The Protocol to Store content is composed by two protocols: the Protocol to Store a Content Item, and the Protocol to Verify the Upload Status of the content item or content element while it is being stored. The content item to store can be either a single MPEG-21 file, or multiple files in case of a DCS, as further explained in the Subclauses below.

The Protocol to Store a Content Item is composed by the following steps.

a) The CCD and the CPD mutually authenticate

b) The CCD generates an mxmscp:TransferProtocolRequest message [8.3.2.3] requesting to store a new content item. This message contains the list of all protocols supported by the CCD (e.g. ftp, http, etc.) which can be employed by the CCD to transfer the content item, including a priority attribute and protocol

options for each protocol. If the CCD supports more protocols than those listed in the mxmscp:StandardProtocol element, these can be expressed by employing the mxmscp:CustomProtocol element.

c) The CPD upon receiving the TransferProtocolRequest message, generates an mxmscp:TransferProtocolResponse message [8.3.2.4] indicating, by means of the Result attribute, if the content can be stored or not. If yes, the ProtocolOptions element will indicate the file transfer protocol selected by the CPD based on the best match between the CCD preferences and the CPD capabilities, including any configuration options that might be necessary for the protocol to be carried out. If not, the ProtocolResult element will convey the reason of failure.

d) In case of an affirmative response from the CPD, the CCD generates an mxmscp:ContentUploadRequest message [8.3.2.5] specifying the information about the content item to store.

1) In the case of a DCF, the following information is specified:

i) The Content ID

ii) The size of the DCF file

iii) The Hash value or digital Signature of the whole DCF file

2) In the case of a DCS, the following information is specified:

i) The choice between

I) The DI – The first time a ContentUploadRequest is sent for storing a DCS, the DCI shall always be specified. It is conceivable, though, that not all the resources that make up a content item shall be stored at the same time. Thus if this approach is taken, starting from the second time the ContentUploadRequest is sent for the same content item, the ContentID shall be sent instead of the DCI.

II) The Content ID

ii) The BBL document specifying how a ContentItem shall be Streamed by employing the Digital Item Streaming technology

iii) A number of mxmscp:Resource elements, which shall be smaller or equal to the number of Resources listed in the DCI. For each Resource the following information shall be specified:

iv) The Resource ID

v) The size of the Resource file

vi) The digital Signature or hash value of the Resource file

e) The CPD, upon receiving the mxmscp:ContentUploadRequest message, generates an mxmscp:ContentUploadResponse message [8.3.2.6] containing the following information:

1) The Result attribute, indicating whether the CCD can start uploading Content or not.

2) In the affirmative case, the StorePath element may indicate one or more remote locations where the file(s) can be uploaded. In the case of multiple locations, the StorePath element specifies the Content ID or Resource ID of the file which should be uploaded at the specified location.

3) In the negative case, the StoreFailure element(s) may indicate the reason of failure for the DCF or for each individual Resource.

f) The CCD:

1) Receives the mxmscp:ContentUploadResponse message from the CPD

2) verifies that it contains an affirmative response, and in this case retrieves from the message the remote location where the file(s) shall be Stored

3) proceeds with transferring the file(s).

Once the file(s) has(have) been transferred, the Protocol to verify the upload status may be employed to verify the status of the upload of a content item or content element. This protocol is composed by the following steps:

a) The CCD and the LPD mutually authenticate

b) The CCD generates an mxmscp:UploadStatusRequest message [8.3.2.7] containing the Identifier of the Content or Content Element whose upload status is sought.

c) The CPD, upon receiving the request, replies with an mxmscp:UploadStatusResponse message [8.3.2.8] containing information related to the status of the upload of the Content Item or Content Element specified in the request.

8.3.2 Protocol data format

This Subclause specifies the payload of the messages exchanged between a Content Creation Device and a Content Provider Device with the purpose of transferring a content item or elements thereof from the former to the latter.

8.3.2.1 StoreContentProtocolType

The mxmscp:StoreContentProtocolType complex type, defined in the figure below, extends the mxmbp:ProtocolType.

```
<complexType name="StoreContentProtocolType" abstract="true">
   <complexContent>
      <extension base="mxmbp:ProtocolType"/>
   </complexContent>
</complexType>
```

Figure 22 — The mxmscp:StoreContentProtocolType complex type

8.3.2.2 Ack

The mxmscp:Ack message, defined in the figure below, is used to acknowledge the success of an operation or to convey an error message in case of failure.

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

Figure 23 — The mxmscp:Ack element

The mxmscp:Ack message extends the mxmbp:ProtocolResult message by adding the “Result” attribute specifying whether the acknowledged operation was successful or otherwise.

8.3.2.3 TransferProtocolRequest

The mxmscp:TransferProtocolRequest message, defined in the figure below, is sent by a CCD to a CPD for requesting the permission to Store a new Content Item in the CPD database/file system.

```
<element name="TransferProtocolRequest"
type="mxmscp:TransferProtocolRequestType"/>
<complexType name="TransferProtocolRequestType">
    <complexContent>
        <extension base="mxmscp:StoreContentProtocolType">
            <sequence>
                <element name="TransferProtocol" type="mxmscp:TransferProtocolType"
maxOccurs="unbounded"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 24 — The mxmscp:TransferProtocolRequest element

This message conveys the list of transfer protocols supported by the CCD which can be used to transfer the file(s). The Signature element may convey an optional digital signature of the message.

The mxmscp:TransferProtocolType complex type defined in the figure below can indicate either a standard protocol, whose list is provided in Table 4 or a custom protocol. Each type of protocol is characterised by a priority attribute, an integer whose value is inversely proportional to the intended priority (priority='1' means top priority).

```
<complexType name="TransferProtocolType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <choice>
                    <element name="StandardProtocol"
type="mxmscp:StandardProtocolType"/>
                    <element name="CustomProtocol"
type="mxmscp:CustomProtocolType"/>
                </choice>
            </sequence>
            <attribute name="priority" type="int" use="required"/>
        </extension>
    </complexContent>
</complexType>
```

Figure 25 — The mxmscp:TransferProtocolType complex type

The mxmscp:StandardProtocolType complex type lists a number of standard protocols which may me employed to transfer the content item or content element file(s). For each of them it is possible to specify options related to the use of the standard protocol (e.g. versions, configuration parameters, etc.).

```
<complexType name="StandardProtocolType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <element name="Option" type="mxmscp:ProtocolOptionType"
minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="ProtocolCode" type="mxmscp:ProtocolCodeType"
use="required"/>
        </extension>
    </complexContent>
</complexType>
<simpleType name="ProtocolCodeType">
    <restriction base="string">
        <enumeration value="FTP"/>
        <enumeration value="HTTP"/>
        <enumeration value="HTTPS"/>
        <enumeration value="OBEX"/>
        <enumeration value="SMB"/>
        <enumeration value="SOAP"/>
        <enumeration value="TCP"/>
        <enumeration value="UDP"/>
    </restriction>
</simpleType>
```

Figure 26 — The mxmscp:StandardProtocolType complex type

The list of standard protocols is given in the table below.

Table 4 — List of standard protocol codes defined in the Store Content Protocol

Protocol Code	Protocol name
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
OBEX	Object Exchange
SMB	Server Message Block
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

The mxmscp:ProtocolOptionType message complex type allows to add parameters to both Standard Protocols and Custom Protocols in a key-value fashion.

```
<complexType name="ProtocolOptionType">
   <simpleContent>
      <extension base="string">
         <attribute name="Key" type="string"/>
      </extension>
   </simpleContent>
</complexType>
```

Figure 27 — The mxmscp:ProtocolOptionType complex type

The mxmscp:CustomProtocolType message enables applications to specify other protocols than those listed in the StandardProtocolType complex type.

```
<complexType name="CustomProtocolType">
   <complexContent>
      <extension base="mxmbp:ProtocolBaseType">
         <sequence>
            <element name="Option" type="mxmscp:ProtocolOptionType"
minOccurs="0" maxOccurs="unbounded"/>
         </sequence>
         <attribute name="ProtocolCode" type="string" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

Figure 28 — The mxmscp:CustomProtocolType complex type

The mxmscp:Ack message extends the mxmbp:ProtocolResult message by adding the “Result” attribute specifying whether the acknowledged operation was successful or otherwise.

8.3.2.4 TransferProtocolResponse

The mxmscp:TransferProtocolResponse message defined in the figure below, is sent by a CPD to a CCD in response to a mxmscp:TransferProtocolRequest message

```
<element name="TransferProtocolResponse"
type="mxmscp:TransferProtocolResponseType"/>
<complexType name="TransferProtocolResponseType">
   <complexContent>
      <extension base="mxmscp:StoreContentProtocolType">
         <sequence>
            <choice>
               <element name="ProtocolResult" type="mxmbp:ProtocolResultType"/>
               <element name="AdoptedProtocol"
type="mxmscp:TransferProtocolType"/>
            </choice>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
         <attribute name="Result" type="boolean" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

Figure 29 — The mxmscp:TransferProtocolResponse element

The response contains a confirmation or a denial of the requested service. In the former case, the Result attribute is set to true, and by means of the ProtocolOptionType complex type, further protocol options may be specified. In the latter case, the Result attribute is set to false and the ProtocolResult element [7.2.4] conveys the reason of failure. The Signature element may convey an optional digital signature of the message.

8.3.2.5 ContentUploadRequest

The mxmscp:ContentUploadRequest message, defined in the figure below, is sent by a CCD to a CPD after receiving a positive mxmscp:TransferProtocolResponse message by the CPD.

```
<element name="ContentUploadRequest" type="mxmscp:ContentUploadRequestType"/>
<complexType name="ContentUploadRequestType">
    <complexContent>
        <extension base="mxmscp:StoreContentProtocolType">
            <sequence>
                <element name="ContentInfo" type="mxmscp:ContentInfoType"
minOccurs="0" maxOccurs="unbounded"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 30 — The mxmscp:ContentUploadRequest element

The mxmscp:ContentUploadRequest message is employed to transmit to the CPD the information about the content to store. The Signature element may convey an optional digital signature of the message.

The mxmscp:ContentInfoType complex type specified in the figure below conveys information related to the content to store, which can be of two types: DCF (the content to store is an MPEG-21 file) or DCS (the content to store is intended to be streamed).

```
<complexType name="ContentInfoType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <choice>
                    <element name="DCF" type="mxmscp:DCFType"/>
                    <element name="DCS" type="mxmscp:DCSType"/>
                </choice>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 31 — The mxmscp:ContentInfoType complex type

An mxmscp:DCFType complex type indicates that the content item is composed by only one file (conforming to ISO/IEC 21000-9).

```
<complexType name="DCFType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <element name="ContentID" type="anyURI"/>
                <element name="Size" type="long"/>
                <element name="ContentSignature" type="dsig:SignatureType"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 32 — The mxmscp:DCFType complex type

The DCFType conveys the following information:

— ContentID: the Content ID

— Size: the size of the DCF file

— ContentSignature: the digital Signature or hash value of the DCF

An mxmscp:DCSType complex type indicates that the content item is composed by multiple files: the Digital Item and a number of referenced content elements.

```
<complexType name="DCSType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <choice>
                    <element ref="didl:DIDL"/>
                    <element name="ContentID" type="anyURI"/>
                </choice>
                <element ref="bbl:BBL" minOccurs="0"/>
                <element name="ResourceInfo" type="mxmscp:ResourceInfoType"
minOccurs="0" maxOccurs="unbounded"/>
                <element name="DIFragment" type="string" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
            <attribute name="DIFragmentsPending" type="boolean" use="optional"
default="false"/>
        </extension>
    </complexContent>
</complexType>
<complexType name="ResourceInfoType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <element name="ResourceID" type="anyURI"/>
                <element ref="bbl:BBL" minOccurs="0"/>
                <element name="Size" type="long"/>
                <element name="ResourceSignature" type="dsig:SignatureType"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 33 — The mxmscp:DCSType complex type

The DCSType conveys the following information:

— A choice between

  — The digital item – The first time this message is sent, the digital item shall be sent. If it is intended not to store all the Resources in a Content Item at the same time, for the subsequent times the digital item may be omitted and replaced with the Content ID

  — The Content ID – When a mxmscp:ContentUploadRequest message for a Content Item has already been sent once, and only additional Resources part of a Content Item shall be Stored

— An optional BBL structure containing the information to be used to stream the DCS

— The list of all Resources intended to be Stored. For each Resource the following information is provided:

  — Resource ID

  — An optional BBL structure containing the information to be used to stream the specific content element part of the DCS

  — Resource Size

  — The digital Resource or Hash value of the Resource.

— An optional DIFragment element conveying portions of a Digital Item to be added to the Digital Item conveyed in the first DCSType according to the Digital Item Streaming standard.

The DCSType has the "DIFragmentsPending" attribute indicating whether Digital Item fragments are expected for a certain DCSType are further expected or else the received DCSType is complete. This attribute is optional and set to "false" by default.

8.3.2.6 ContentUploadResponse

The mxmscp:ContentUploadResponse message, defined in the figure below, is sent by a CPD to a CCD in response to an mxmscp:ContentUpload Request message.

```
<element name="ContentUploadResponse" type="mxmscp:ContentUploadResponseType"/>
<complexType name="ContentUploadResponseType">
   <complexContent>
      <extension base="mxmscp:StoreContentProtocolType">
         <sequence>
            <element name="EntityResult" type="mxmscp:EntityResultType"
maxOccurs="unbounded"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
         <attribute name="Result" type="boolean" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

Figure 34 — The mxmscp:ContentUploadResponse element

The mxmscp:ContentUploadResponse message is employed to transmit the result of the request to store a content item. In the case the content item to upload specified in the ContentUploadRequest message is of type DCF, the ContentUploadResponse shall contain one and only one EntityResult, while if the content to store is of type DCS, the ContentUploadResponse shall contain as many EntityResult as the number of mxmscp:ResourceInfo specified in the request. The Signature element may convey an optional digital signature of the message.

In case of success, the StoragePath element contains the information required to store the corresponding content item or DCS element: the URL of the server/repository where the file shall be stored and a number of options which may be required (e.g. username, password, etc.), while in case of failure the StorageFailure element contains an error message.

```
<complexType name="EntityResultType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <element name="ContentID" type="anyURI"/>
                <element name="ResourceID" type="anyURI" minOccurs="0"/>
                <choice>
                    <element name="StoragePath" type="mxmscp:StoragePathType" />
                    <element name="StorageFailure" type="mxmbp:ProtocolResultType"
/>
                </choice>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="StoragePathType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <element name="StorageEntry" type="anyURI"/>
                <element name="EntryInfo" type="mxmscp:ProtocolOptionType"
minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 35 — The mxmscp:EntityResultType complexType

8.3.2.7 UploadStatusRequest

The mxmscp:UploadStatusRequest defined in the figure below, allows a CCD to query for the status of a Content Item or Content Element in the process of being uploaded.

```
<element name="UploadStatusRequest" type="mxmscp:UploadStatusRequestType"/>
<complexType name="UploadStatusRequestType">
    <complexContent>
        <extension base="mxmscp:StoreContentProtocolType">
            <sequence>
                <element name="ContentID" type="anyURI"/>
                <element name="ResourceID" type="anyURI" minOccurs="0"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 36 — The mxmscp:UploadStatusRequest element

In the request message it is possible to specify the specific Content Item or Content Element Identifier whose upload status is the target of the request. The Signature element may convey an optional digital signature of the message.

8.3.2.8 UploadStatusResponse

The mxmscp:UploadStatusResponse message, defined in the figure below, conveys the response from a CPD to the CCD related to a mxmscp:UploadStatusRequest about the status of a Content Item or Content Element queried during or after the phase of uploading.

```
<element name="UploadStatusResponse" type="mxmscp:UploadStatusResponseType"/>
<complexType name="UploadStatusResponseType">
   <complexContent>
      <extension base="mxmscp:StoreContentProtocolType">
         <sequence>
            <element name="UploadStatus" type="mxmscp:UploadStatusType"/>
            <element name="DisplayString" type="string" minOccurs="0"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
<simpleType name="UploadStatusType">
   <restriction base="string">
      <enumeration value="TO_BE_UPLOADED"/>
      <enumeration value="CURRENTLY_IN_UPLOAD"/>
      <enumeration value="SUCCESSFULLY_UPLOADED"/>
      <enumeration value="UNSUCCESSFULLY_UPLOADED"/>
      <enumeration value="CONTENT_UNKNOWN"/>
      <enumeration value="NOT_SPECIFIED"/>
   </restriction>
</simpleType>
```

Figure 37 — The mxmscp:UploadStatusResponse element

The response message contains the information about the status of the content item or content element file in the process of, or after being uploaded. The mxmscp:UploadStatusType element conveys the information about upload status, while the DisplayString element conveys a textual description of the upload status. The Signature element may convey an optional digital signature of the message

The list of standard protocols is given in the table below.

Table 5 — Possible values of the UploadStatusType simple type

Protocol Code	Protocol name
TO_BE_UPLOADED	The server is waiting for the upload of the specified content item or content element, which has not been initiated yet
CURRENTLY_IN_UPLOAD	The specified content item or content element is currently being uploaded
SUCCESSFULLY_UPLOADED	The specified content item or content element has been successfully uploaded
UNSUCCESSFULLY_UPLOADED	The specified content item or content element has not been successfully uploaded, and has to be re-uploaded
CONTENT_UNKNOWN	There is no record for the specified content item or content element on the server
NOT_SPECIFIED	The server is unable to respond to the request for an unknown reason

8.4 Access Content Protocol

8.4.1 Protocol specification

This Subclause 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 protocol involves the following steps:

a) The requesting device and the Content Provider Device mutually authenticate
b) The requesting device:
    1) generates a mxmacp:RequestContent message [8.4.2.3].
    2) optionally signs the message.
    3) sends the mxmacp: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 mxmacp:RequestContentResponse message [8.4.2.4] containing:
    1) the digital item representing the requested content if the mxmacp: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 mxmacp: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 mxmacp:ContentElementID parameter in the request can be retrieved, in the case the mxmacp: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 mxmacp:Ack message [8.4.2.2] conveying the information related to the reason of failure.
f) The requesting Device:
    1) (optionally) replies with a mxmacp:Ack message [8.4.2.2]
    2) retrieves the content from the location specified in the mxmacp:RequestContentResult message and uses it according to the license terms

8.4.2 Protocol data format

This Subclause specifies the payload of the messages employed by a Device (e.g. an End-User Device) to access a Content Item or parts thereof from a Content Provider Device.

8.4.2.1 AccessContentProtocolType

The mxmacp:AccessContentProtocolType complex type, defined in the figure below, extends the mxmbp:ProtocolType.

```
<complexType name="AccessContentProtocolType" abstract="true">
   <complexContent>
      <extension base="mxmbp:ProtocolType"/>
   </complexContent>
</complexType>
```

Figure 38 — The mxmacp:AccessContentProtocolType complex type

8.4.2.2 Ack

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

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

Figure 39 — The mxmacp:Ack element

8.4.2.3 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="mxmacp:RequestContentType"/>
<complexType name="RequestContentType">
    <complexContent>
        <extension base="mxmacp:AccessContentProtocolType">
            <sequence>
                <element name="ContentIdentifier"
type="mxmbp:ContentIdentifierType"/>
                <element name="MimeType" type="string" minOccurs="0"/>
                <element ref="rel-r:license" minOccurs="0"/>
                <element name="UsageEnvironmentDescription"
type="dia:UsageEnvironmentType" minOccurs="0"/>
                <element name="UniversalConstraintDescription" type="dia:UCDType"
minOccurs="0" maxOccurs="unbounded"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 40 — The mxmacp:RequestContent element

The mxmacp:RequestContent message conveys the following information:

— mxmbp:ContentIdentifier: the identifier of the requested content item or content element within a content item, as defined in 7.2.5

— mxmacp: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 mxmbp:ContentIdentifier is requested

  — the mime type of the resource identified in mxmbp:ContentIdentifier

— r:license: an optional license specifying additional information about the requested license needed to access the requested content

— mxmacp:UsageEnvironmentDescription: an optional element specifying the usage environment of the device on which content will be used. 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

— mxmacp:UniversalConstraintDescription: an optional element which may be used to further constrain the usage of a digital item.

— dsig:Signature: an optional digital signature of the mxmacp: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 mxmacp:ContentElementIdentifier element is specified in an mxmacp:RequestContent message, this implies that only the specific content element is requested, and not the whole content item.

8.4.2.4 RequestContentResponse

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

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

Figure 41 — The mxmacp:RequestContentResponse element

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

— mxmacp:DI: the optional digital item representing the content item being requested

— mxmacp: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

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 mxmacp:ContentURL elements shall be returned in the mxmacp:RequestContentResponse, one indicating the URL for the audio elementary stream and the other the URL for the metadata elementary stream.

The mxmacp:ContentURLType complex type is specified in the figure below:

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

Figure 42 — The mxmacp:RequestContentResponse element

9 License Protocols

9.1 Store License Protocol

The Store License Protocol is typically performed between a Content Creation Device (CCD) and a License Provider Device (LPD). The scope of this protocol is to send a license to the LPD granting it the right to issue licenses for a specific content item to other devices upon request.

9.1.1 Protocol specification

The Protocol to Store License involves the following steps:

a) The CCD and the LPD mutually Authenticate

b) The CCD generates an mxmslp:StoreLicenseRequest message [9.1.2.3] containing:

   1) the mxmslp:ContentItem element of type mxmslp:ItemType and conveying the Rights information for the whole Content Item

   2) a number of mxmslp:ContentElement elements of type mxmslp:ItemType and conveying the Rights information for every Governed Content Element.

   3) the digital Signature for the whole message.

c) The License Provider Device, upon receiving the request message, if the request can be satisfied,

   1) verifies the digital Signature;

   2) Stores the Rights information related to the Content Item and the Content Elements into a persistent location, so that every time a device (e.g. an End-User Device) performs the Protocol to Access License specifying either the Content ID or the Content ID together with the ContentElementID, the LPD will have the information required at its disposal and may determine whether the request can be satisfied or not..

   3) sends an mxmslp:Ack message [9.1.2.2] back to CCD, indicating the result of the Store License.

9.1.2 Protocol data format

This Subclause specifies the payload of the messages part of the Store License Protocol.

9.1.2.1 StoreLicenseProtocolType

The mxmslp:StoreLicenseProtocolType complex type, defined in the figure below, extends the mxmbp:ProtocolType.

```
<complexType name="StoreLicenseProtocolType" abstract="true">
   <complexContent>
      <extension base="mxmbp:ProtocolType"/>
   </complexContent>
</complexType>
```

Figure 43 — The mxmslp:StoreLicenseProtocolType complex type

9.1.2.2 Ack

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

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

Figure 44 — The mxmslp:Ack element

9.1.2.3 StoreLicenseRequest

The mxmslp:StoreLicenseRequest message, defined in the figure below, is used to send one or rights information related to a content item and any of its governed content elements the LPD.

```
<element name="StoreLicenseRequest" type="mxmslp:StoreLicenseRequestType"/>
<complexType name="StoreLicenseRequestType">
   <complexContent>
      <extension base="mxmslp:StoreLicenseProtocolType">
         <sequence>
            <element name="ContentItem" type="mxmslp:ItemType"/>
            <element name="ContentElement" type="mxmslp:ItemType" minOccurs="0"
maxOccurs="unbounded"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 45 — The mxmslp:StoreLicenseRequest element

A mxmslp:StoreLicenseRequest message conveys the information related to the rights governing a Content Item, conveyed by the mxmslp:ContentItem element and those related to any of its content elements, conveyed by the mxmslp:ContentElement element. In the case the StoreLicense message is digitally signed, the digital signature shall be inserted in the dsig:Signature element.

```
<complexType name="ItemType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <element name="ID" type="anyURI"/>
                <element name="License" type="rel-r:License" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 46 — The mxmslp:ItemType complex type

An mxmslp:ItemType complexType conveys the following information:

— ID: either the Content ID or a Content Element ID, depending on whether this complex type is used by an xmxslp:ContentItem or mxmslp:ContentElement

— License: an ISO/IEC 21000-5-compliant license granting to the LPD the right to issue licenses to other devices upon request, possibly under some conditions.

9.2 Revoke License Protocol

The Revoke License Protocol is typically performed between a Content Creation Device (CCD) and a License Provider Device (LPD). The scope of this protocol is to revoke a license previously stored on the LPD.

9.2.1 Protocol specification

The Protocol to Revoke a License involves the following steps:

a) The CCD and the LPD mutually Authenticate

b) The CCD generates an mxmrlp:RevokeLicenseRequest message [9.2.2.3] containing:

1) the identifier(s) of the license(s) previously stored on the LPD, which shall be revoked

2) an optional date from when the specified licenses shall be revoked. If this element is omitted the licenses will be revoked as soon as the message is received.

3) an optional digital Signature for the whole message.

c) The License Provider Device, upon receiving the request message, if the request can be satisfied,

1) verifies the digital Signature;

2) verifies whether the specified licenses indeed exist and acts in order to revoke them

3) sends an mxmrlp:Ack message [9.2.2.2] back to CCD, indicating the result of the Revoke License protocol.

9.2.2 Protocol data format

This Subclause specifies the payload of the messages part of the Revoke License Protocol.

9.2.2.1 RevokeLicenseProtocolType

The mxmrlp:RevokeLicenseProtocolType complex type, defined in the figure below, extends the mxmbp:ProtocolType.

```
<complexType name="RevokeLicenseProtocolType" abstract="true">
    <complexContent>
        <extension base="mxmbp:ProtocolType"/>
    </complexContent>
</complexType>
```

Figure 47 — The mxmrlp:RevokeLicenseProtocolType complex type

9.2.2.2 Ack

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

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

Figure 48 — The mxmrlp:Ack element

9.2.2.3 RevokeLicenseRequest

The mxmrlp:RevokeLicenseRequest message, defined in the figure below, is used to revoke one or more licenses previously stored on the LPD.

```
<element name="RevokeLicenseRequest" type="mxmrlp:RevokeLicenseRequestType"/>
<complexType name="RevokeLicenseRequestType">
    <complexContent>
        <extension base="mxmrlp:RevokeLicenseProtocolType">
            <sequence>
                <element name="LicenseID" type="anyURI" maxOccurs="unbounded"/>
                <element name="DateOfRevocation" type="dateTime" minOccurs="0"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 49 — The mxmrlp:RevokeLicenseRequest element

An mxmrlp:RevokeLicenseRequest element conveys the following information:

— LicenseID: the identifier of the licenses previously stored that shall be revoked

— DateOfRevocation: the date and time from which the specified license(s) shall be revoked. If this element is not specified, the revocation of the specified licenses shall be carried out immediately after the message is received.

— Signature: an optional digital signature of the message.

9.3 Access License Protocol

The Access License Protocol is performed by a device to access licenses from a License Provider Device.

9.3.1 Protocol specification

The Access License Protocol 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 mxmalp:RequestLicense message [9.3.2.3] containing the following elements:
   1) a choice between either
      i. the identifier or the content item or content element for which a license is sought
      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 mxmalp: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)
      ii. a valid license identifier is specified in the request
   3) verifies whether
      i. there is a license on the system
      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 mxmalp:RequesLicenseResponse message [9.3.2.4] containing the license. In the case the license grants rights over a protected resource, the license may 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 mxmalp:RequesLicenseResult message to the requesting Device:
g) In the case the request cannot be satisfied, the License Provider Device generates a mxmalp:RequesLicenseResponse message [9.3.2.4] containing a false value in the Result attribute and specifies the reason of failure in the ErrorCode field.
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 mxmalp:Ack message [9.3.2.2] to the License Provider Device signalling that the response was successfully received.

9.3.2 Protocol data format

This Subclause specifies the payload of the messages part of the Store License Protocol.

9.3.2.1 AccessLicenseProtocolType

The mxmalp:AccessLicenseProtocolType complex type, defined in the figure below, extends the mxmbp:ProtocolType.

```
<complexType name="AccessLicenseProtocolType" abstract="true">
   <complexContent>
      <extension base="mxmbp:ProtocolType"/>
   </complexContent>
</complexType>
```

Figure 50 — The mxmalp:AccessLicenseProtocolType complex type

9.3.2.2 Ack

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

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

Figure 51 — The mxmalp:Ack element

9.3.2.3 Request License

The mxmalp: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 mxmalp: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="mxmalp:RequestLicenseType"/>
<complexType name="RequestLicenseType">
   <complexContent>
      <extension base="mxmalp:AccessLicenseProtocolType">
         <sequence>
            <choice>
               <element name="ContentIdentifier"
type="mxmbp:ContentIdentifierType"/>
               <element name="LicenseID" type="anyURI"/>
            </choice>
            <element ref="rel-r:license" minOccurs="0"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 52 — The mxmalp:RequestLicense element

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

— mxmalp:ContentIdentifier: the identifier of the governed asset for which a license is requested

— mxmalp: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 mxmalp:RequestLicense message.

9.3.2.4 RequestLicenseResponse

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

```
<element name="RequestLicenseResponse" type="mxmalp:RequestLicenseResponseType"/>
<complexType name="RequestLicenseResponseType">
    <complexContent>
        <extension base="mxmalp:AccessLicenseProtocolType">
            <sequence>
                <choice>
                    <element ref="rel-r:license"/>
                    <element name="ErrorCode" type="mxmbp:ResultCodeType"/>
                </choice>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
            <attribute name="Result" type="boolean" use="required"/>
        </extension>
    </complexContent>
</complexType>
```

Figure 53 — The mxmalp:RequestLicenseResponse element

The mxmalp: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.

10 IPMP Tool Protocols

10.1 Access IPMP Tool Protocol

This protocol is performed by a device (e.g. an End-User Device) to retrieve an IPMP Tool Body from an IPMP Tool Provider Device.

10.1.1 Protocol specification

This protocol is carried out in the following steps:

a) The EUD 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 EUD sends an mxmaitp:RequestIPMPToolBody message [10.1.2.3] 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) either delivers the ToolBody as part of the mxmaitp:RequestIPMPToolBodyResponse [10.1.2.4] to the EUD, or inserts in the mxmaitp:RequestIPMPToolBodyResponse the location from where the ToolBody can be retrieved.

d) The EUD
   1) receives the mxmaitp:RequestIPMPToolBodyResponse message
   2) unpackages the ipmpinfo-msx:ToolBody
   3) uses the IPMP Tool Body

10.1.2 Protocol data format

This Subclause specifies the payload of the messages part of the Access IPMP Tool Protocol.

10.1.2.1 AccessIPMPToolProtocolType

The mxmaitp:AccessIPMPToolProtocolType complex type, defined in the figure below, extends the mxmbp:ProtocolType.

```
<complexType name="AccessIPMPToolProtocolType" abstract="true">
   <complexContent>
      <extension base="mxmbp:ProtocolType"/>
   </complexContent>
</complexType>
```

Figure 54 — The mxmaitp:AccessIPMPToolProtocolType complex type

10.1.2.2 Ack

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

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

Figure 55 — The mxmaitp:Ack element

10.1.2.3 RequestIPMPToolBody

The mxmaitp: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="mxmaitp:RequestIPMPToolBodyType"/>
<complexType name="RequestIPMPToolBodyType">
   <complexContent>
      <extension base="mxmaitp:IPMPToolProtocolType">
         <sequence>
            <element ref="ipmpinfo:IPMPToolID"/>
            <element ref="ipmpinfo-msx:DeviceInformation"/>
            <element ref="dsig:Signature" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figure 56 — The mxmaitp:RequestIPMPToolBody element

The mxmaitp: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.

— dsig:Signature: shall this message be signed, the digital signature shall be included in the dsig:Signature element.

10.1.2.4 RequestIPMPToolBodyResponse

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

```
<element name="RequestIPMPToolBodyResponse"
type="mxmaitp:RequestIPMPToolBodyResponseType"/>
<complexType name="RequestIPMPToolBodyResponseType">
    <complexContent>
        <extension base="mxmaitp:IPMPToolProtocolType">
            <sequence>
                <choice maxOccurs="unbounded">
                    <element ref="ipmpinfo-msx:ToolBody"/>
                    <element name="ToolURL" type="anyURI"/>
                </choice>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 57 — The mxmaitp:RequestIPMPToolBodyResponse element

The mxmaitp: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.

10.2 Access IPMP Tool List Protocol

This Protocol is employed to request the list of IPMP Tool bodies present on an IPMP Tool Provider Device so that a decision can be made in order to select and retrieve an appropriate IPMP Tool Body and use it to protect a content item or parts thereof.

10.2.1 Protocol specification

This protocol is always started by the requesting Device, and it is as follows:

a) The requesting Device (e.g. a CCD) and the TPD mutually Authenticate.
b) The requesting Device sends to the TPD an mxmaitp:RequestIPMPToolInfoList message requesting the TPD the list of IPMP Tool Bodies satisfying some criteria.
c) The TPD replies with an mxmaitp:RequestIPMPToolInfoListResponse conveying to the requesting Device the list of IPMP Tool body satisfying the criteria in the request, so the requesting device can select an appropriate IPMP Tool Body and Access it using the mxmaitp:RequestIPMPToolBody request message.

10.2.2 Protocol data format

10.2.2.1 RequestIPMPToolInfoList

The mxmaitp:RequestIPMPToolInfoList message is employed to request a TPD the list of IPMP Tool Body information items available on the Device.

```
<element name="RequestIPMPToolInfoList"
type="mxmaitp:RequestIPMPToolInfoListType"/>
<complexType name="RequestIPMPToolInfoListType">
    <complexContent>
        <extension base="mxmaitp:IPMPToolProtocolType">
            <sequence>
                <element ref="ipmpinfo-msx:DeviceInformation" minOccurs="0"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 58 — The mxmaitp:RequestIPMPToolInfoList element

The RegisterERR message may contain an optional ipmpinfo-msx:DeviceInformation element as defined in [18] specifying the characteristics of the platform on which the IPMP Tool shall operate, and an optional Digital Signature for message integrity.

10.2.2.2 RequestIPMPToolInfoListResponse

The mxmaitp:RequestIPMPToolInfoListResponse message is sent in response to an mxmaitp:RequestIPMPToolInfoList message and conveys the list of IPMP Tool Body information items available on the Device.

```
<element name="RequestIPMPToolInfoListResponse"
type="mxmaitp:RequestIPMPToolInfoListResponseType"/>
<complexType name="RequestIPMPToolInfoListResponseType">
    <complexContent>
        <extension base="mxmaitp:IPMPToolProtocolType">
            <sequence>
                <element ref="mxmaitp:IPMPToolInfo" minOccurs="0"
maxOccurs="unbounded"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<element name="IPMPToolInfo" type="mxmaitp:IPMPToolInfoType"/>
<complexType name="IPMPToolInfoType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <element ref="ipmpinfo:IPMPToolID"/>
                <element ref="ipmpmsg:ParametricDescription" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 59 — The mxmaitp: RequestIPMPToolInfoListResponse element

The RequestIPMPToolInfoListResponse message may contain any number of mxmaitp:IPMPToolInfo elements, each representing an IPMP Tool Body available on the TPD satisfying the criteria specified in the request. An mxmaitp:IPMPToolInfo element conveys the IPMP Tool identifier of the IPMP Tool and a parametric description of the IPMP Tool so the requesting Device may select the most suitable IPMP Tool body. The message may also contain an optional Digital Signature for message integrity.

11 Domain management protocols

11.1 Introduction

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

11.2 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.

The figure below represents a possible domain configuration with:

The figure below 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.

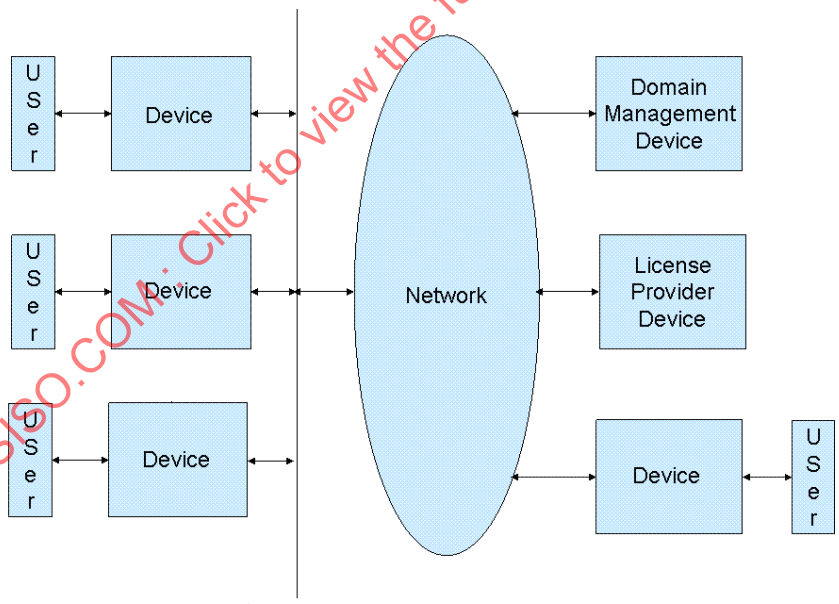

Figure 60 — An example of domain

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:

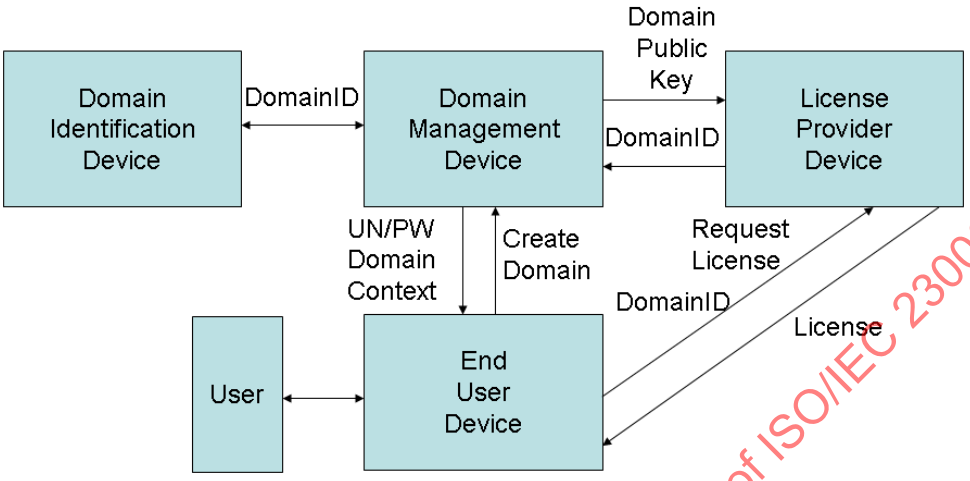

Figure 61 — Some of the relationship of the 5 Device types in Manage Domain

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.

11.3 Domain Information specification

This Subclause specifies the information associated with domain management.

11.3.1 Common elements defined in the mxmd namespace

The domain namespace defines the following elements:

```
<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="mxmd:IDType"/>
<element name="DeviceID" type="mxmd:IDType"/>
<element name="LocalDomainID" type="mxmd: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 62 — Common mxmd elements

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

- `mxmd:DomainManagerID`: the identifier assigned to a Domain Manager Device (DMD) by an appropriate registration authority
- `mxmd:AccessID`: the Access ID given to the Domain Administrator
- `mxmd:AccessPassword`: the Access Password given to the Domain Administrator
- `mxmd:DomainKey`: the decryption key for Domain-encrypted content key.
- `mxmd:UserID`: the identifier associated with a user, of type mxmd:IDType (See 11.3.3)
- `mxmd:DeviceID`: the Identifier associated with a device, of type mxmd:IDType (See 11.3.3)
- `mxmd:LocalDomainID`: the Local Domain Identifier issued by the Domain Identification Device
- `mxmd:ContentGroupID`: the Identifier assigned to a group of content items for the management of domains
- `mxmd:MaximumNumberOfDevices`: the maximum number of devices allowed in a domain
- `mxmd:MaximumNumberOfUsers`: the maximum number of users allowed in a domain
- `mxmd:MaximumFrequencyOfUpdateDevice`: the minimum time interval before a device that has left a domain may be allowed to re-join it
- `mxmd:MaximumFrequencyOfUpdateUser`: the minimum time interval before a user who has left a domain may be allowed to re-join it
- `mxmd:Expiration` : The time at which the domain expires

11.3.2 DomainBaseType

The MXM Domain namespace, mxmd, 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 63 — The mxmd:DomainBaseType complex type

11.3.3 IDType

The mxmd: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 64 — The mxmd:IDType complex type

The mxmd: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 mxmd:id element, or an X.509 certificate, in which case it shall be expressed according to the dsig:X509Data element and conformant to [10].

11.3.4 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 11.6.2. The result of this protocol is the creation by the DMD of a mxmd:DomainManageInfo element specified in the figure below:

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

Figure 65 — The mxmd:DomainManageInfo element

In the above, the following semantics apply:

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

11.3.5 DACredentials and DomainMembershipCredentials

The mxmd: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 mxmd:DomainMembershipCredentials are credentials employed by a device or a user to authenticate with a DMD.

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

Figure 66 — The mxmd:DACredentials and mxmd:DomainMembershipCredentials elements

11.3.6 DomainID

The mxmd:DomainID element is defined as follows:

```
<element name="DomainID" type="mxmd:DomainIDType"/>
<complexType name="DomainIDType">
    <complexContent>
```

```
        <extension base="mxmd:IDType">
            <sequence>
                <element ref="mxmd:DomainManagerID"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 67 — The mxmd:DomainID element

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

11.3.7 User

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

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

Figure 68 — The mxmd:User element

The mxmd: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, mxmd:UserIDList, as defined in the figure below
- the mxmd: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 mxmd:UserIDList element is defined in the Figure below:

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

Figure 69 — The mxmd: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.

11.3.8 Device

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

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

Figure 70 — The mxmd:Device element

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

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

The mxmd:DeviceIDList element is defined in the figure below:

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

Figure 71 — The mxmd:DeviceIDList element

The mxmd: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.

11.4 Domain Use Data specification

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

11.4.1 UseData

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

```
<element name="UseData" type="mxmd:UseDataType"/>
<complexType name="UseDataType">
    <sequence>
        <element ref="mxmd:DomainID"/>
        <element ref="mxmd:Record" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

Figure 72 — The mxmdp:UseData element

11.4.2 Record

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

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

Figure 73 — The mxmdp:Record element

The semantics for `mxmdp: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.

11.5 Domain Protocol Information specification

This Subclause 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 Subclause include a number of elements defined in the mxmd namespace specified in 11.3, and new elements in the MXM Domain Protocol namespace, mxmdp.

11.5.1 DomainProtocolType

The mxmdp:DomainProtocolType complex type, defined in the figure below, extends the mxmbp:ProtocolType.

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

Figure 74 — The mxmdp: DomainProtocolType complex type

11.5.2 Ack

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

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

Figure 75 — The mxmdp:Ack element

11.5.3 AuthenticateReq

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

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

Figure 76 — The mxmdp: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.

11.5.4 LocalDomainIDRequest

The mxmdp: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="mxmdp:RequestLocalDomainIDType"/>
<complexType name="RequestLocalDomainIDType">
   <complexContent>
      <extension base="mxmdp:DomainProtocolType"/>
   </complexContent>
</complexType>
```

Figure 77 — The mxmdp:LocalDomainIDRequest element

11.5.5 LocalDomainIDResponse

The mxmdp: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="mxmdp:LocalDomainIDResponseType"/>
<complexType name="LocalDomainIDResponseType">
    <complexContent>
        <extension base="mxmdp:DomainProtocolType">
            <sequence>
                <element ref="mxmd:LocalDomainID"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 78 — The mxmdp:LocalDomainIDResponse element

11.5.6 RequestKey

The mxmdp: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="mxmdp:RequestKeyType"/>
<complexType name="RequestKeyType">
    <complexContent>
        <extension base="mxmdp:DomainProtocolType">
            <sequence>
                <element ref="mxmd:DomainID"/>
                <element ref="mxmd:ContentGroupID" minOccurs="0"
maxOccurs="unbounded"/>
                <choice minOccurs="0" maxOccurs="unbounded">
                    <element ref="mxmd:DeviceID"/>
                    <element ref="mxmd:UserID"/>
                </choice>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 79 — The mxmdp:RequestKey element

The semantics for the mxmdp: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

11.5.7 RequestKeyResponse

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

```
<element name="RequestKeyResponse" type="mxmdp:RequestKeyResponseType"/>
<complexType name="RequestKeyResponseType">
    <complexContent>
        <extension base="mxmdp:DomainProtocolType">
            <sequence>
                <element ref="mxmd:DomainKey"/>
                <element ref="mxmd:UserID" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="mxmd:DeviceID" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 80 — The mxmdp:RequestKeyResponse element

The semantics for mxmdp:RequestKeyResponse message is given below:

- mxmd:DomainKey: element used to convey the encryption key of the domain indicated by DomainID in the preceding mxmdp: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.
- mxmd:DeviceID: the identifiers of the devices members of the domain identified by the mxmd:DomainID specified in the preceding mxmdp:RequestKey.
- mxmd:UserID: the identifiers of the users members of the domain indicated by mxmd:DomainID specified in the preceding mxmdp:RequestKey.

11.5.8 AddDevice

The mxmdp:AddDevice message specified in the figure below is sent by an MXM Device to a DMD requesting to join a domain.

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

Figure 81 — The mxmdp:AddDevice element

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

11.5.9 AddUser

The mxmdp:AddUser message specified in the figure below is sent by an MXM Device to a DMD requesting its user to become a member of a domain.

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

Figure 82 — The mxmdp:AddUser element

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

11.5.10 RenewDevice

The mxmdp:RenewDevice message specified in the figure below is sent by an MXM Device to a DMD requesting its membership to the domain to be renewed.

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

Figure 83 — The mxmdp:RenewDevice element

This message conveys the identifier of the device requesting the renewal of domain membership, and the use data (see 11.7.2) generated while using content bound to that domain.

11.5.11 RenewUser

The mxmdp:RenewUser message specified in the figure below is sent by an MXM Device to a DMD requesting the membership of its user to the domain to be renewed.

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

Figure 84 — The mxmdp:RenewUser element

This message conveys the identifier of the user requesting the renewal of domain membership, and the use data (see 11.7.2) generated while using content bound to that domain.

11.5.12 AddDeviceResponse, AddUserResponse, RenewDeviceResponse and RenewUserResponse

The four messages defined in the figure below are sent in response to 11.5.8, 11.5.9, 11.5.10 and 11.5.11 respectively.

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

Figure 85 — The mxmdp:AddDeviceResponse, mxmdp:AddUserResponse, mxmdp:RenewDeviceResponse, and mxmdp: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 86 — An example of domain membership license

11.5.13 LeaveDevice

The mxmdp:LeaveDevice message is sent by an MXM Device 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="mxmdp:LeaveDeviceType"/>
<complexType name="LeaveDeviceType">
    <complexContent>
        <extension base="mxmdp:DomainProtocolType">
            <sequence>
                <element ref="mxmd:DeviceID"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 87 — The mxmdp:LeaveDevice element

11.5.14 LeaveUser

The mxmdp:LeaveUser message is sent by the user of an MXM Device 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="mxmdp:LeaveUserType"/>
<complexType name="LeaveUserType">
    <complexContent>
        <extension base="mxmdp:DomainProtocolType">
            <sequence>
                <element ref="mxmd:UserID"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 88 — The mxmdp:LeaveUser element

11.5.15 CreateDomain

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

```
<element name="CreateDomain" type="mxmdp:CreateDomainType"/>
<complexType name="CreateDomainType">
    <complexContent>
        <extension base="mxmdp:DomainProtocolType">
            <sequence>
                <element ref="mxmd:DACredentials"/>
                <element ref="mxmd:Expiration"/>
                <element ref="mxmd:MaximumNumberOfUsers" minOccurs="0"/>
                <element ref="mxmd:MaximumNumberOfDevices" minOccurs="0"/>
                <element ref="mxmd:MaximumFrequencyOfUpdateUser" minOccurs="0"/>
                <element ref="mxmd:MaximumFrequencyOfUpdateDevice" minOccurs="0"/>
                <element ref="mxmd:UserRevocationList" minOccurs="0"/>
                <element ref="mxmd:DeviceRevocationList" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 89 — The mxmdp: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.

11.5.16 CreateDomainResponse

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

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

Figure 90 — The mxmdp:CreateDomainResponse element

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

11.5.17 RenewDomain

The mxmdp: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="mxmdp:RenewDomainType"/>
<complexType name="RenewDomainType">
    <complexContent>
        <extension base="mxmdp:DomainProtocolType">
            <sequence>
                <element ref="mxmd:Expiration"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 91 — The mxmdp:RenewDomain element

This message conveys the expiration information for the renewed domain.

11.5.18 DeleteDomain

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

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

Figure 92 — The mxmdp:DeleteDomain element

11.5.19 UnLicensedSimultaneousUseNotice

The mxmdp:UnLicensedSimultaneousUseNotice message specified in the figure below is sent by an MXM Device to the DMD for communicating irregularities in the use of domain-bound content.

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

Figure 93 — The mxmdp:UnLicensedSimultaneousUseNotice element

A detailed explanation of the use of the mxmdp:UnlicensedSimultaneousUseNotice element is given in 11.7.

11.6 Domain Management Protocols specification

11.6.1 Introduction

This Subclause specifies how to employ the messages defined in 11.5 to manage domains. The functionalities of these protocols include:

- setting up a domain described by domain information represented in mxmd: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.

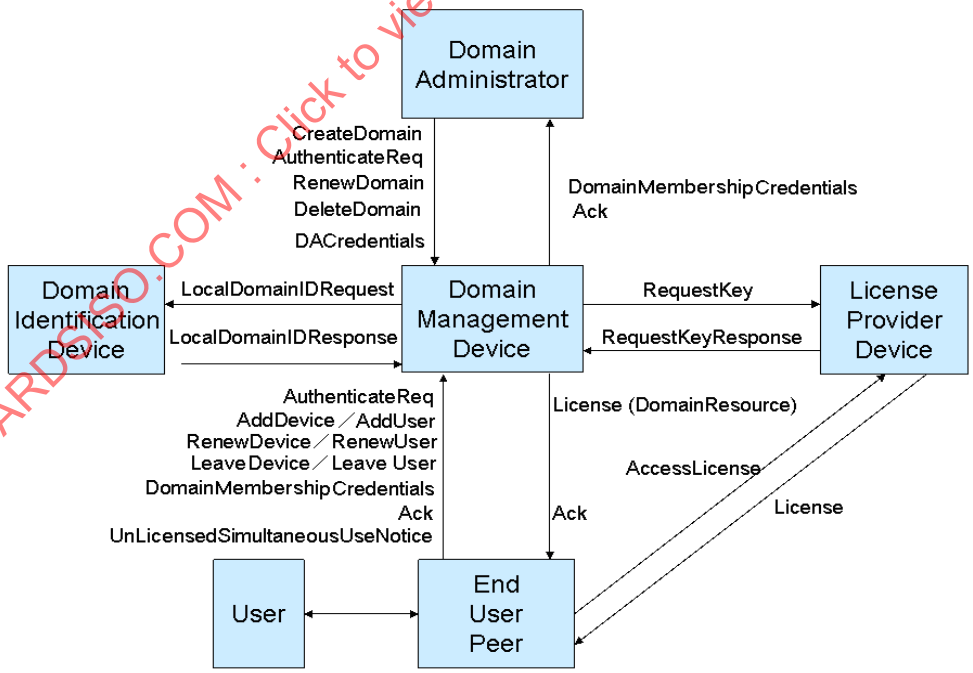

Figure 94 — Schematic Overview of Manage Domain

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 Subclauses.

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 mxmd:LocalDomainID and sends it to the DMD
d) The DMD:
    1) generates the mxmd:DomainID.
    2) creates the domain information, mxmd: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 (mxmd:DeviceID) or user identifier (mxmd:UserID) to the corresponding device or user list in mxmd: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.

11.6.2 Protocols between the Domain Administrator and the DMD

This Subclause specifies the following protocols

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

11.6.2.1 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 mxmdp:CreateDomain message [Figure 89] 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 mxmd:LocalDomainID [Figure 62] by sending a mxmdp:LocalDomainIDRequest message [Figure 77]
d) If the request can be satisfied, the DID sends an mxmdp:LocalDomainIDResponse [Figure 78] to the DMD
e) The DMD
    1) creates a number of mxmd:DomainMembershipCredentials to be given to all Devices and Users allowed to join the domain
    2) generates the domain key, mxmd:DomainKey
    3) generates and sends the mxmdp:CreateDomainResponse message [Figure 90] to the Domain Administrator or the mxmdp:Ack message [Figure 75] to acknowledge an unsuccessful completion of the Protocol.
f) the Domain Administrator:
    1) stores the mxmd:DomainMembershipCredentials

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

11.6.2.2 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 mxmd:DomainManageInfo.

a) Domain Administrator and DMD mutually Authenticate
b) Domain Administrator issues mxmdp:AuthenticateReq [Figure 76], identifying the domain to be renewed.
c) DMD returns mxmdp:Ack [Figure 75].
d) Domain Administrator issues mxmdp:RenewDomain [Figure 91].
e) DMD
    1) renews mxmd:DomainManageInfo
    2) (optionally) returns an mxmdp:Ack [Figure 75] acknowledging the success of the Protocol or otherwise.

11.6.2.3 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 mxmdp:AuthenticateReq [Figure 76] , identifying the domain to be deleted.
c) DMD returns mxmdp:Ack [Figure 75].
d) Domain Administrator issues DeleteDomain [Figure 92]
e) DMD deletes Domain
f) (optionally) DMD returns mxmdp:Ack [Figure 75] acknowledging the success of the protocol or otherwise.

11.6.3 Protocol between the DMD and the LPD

11.6.3.1 DMD-LPD protocol

This Subclause 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 mxmdp:RequestKey [Figure 79]
c) DMD delivers the key to the LPD using mxmdp:RequestKeyResponse [Figure 80]
d) (optionally) LPD replies with mxmdp:Ack [Figure 75].

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

11.6.4 Protocols between the device/User and the DMD

This Subclause 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

11.6.4.1 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 mxmd:Device part of the mxmd:DomainManageInfo. The protocol for adding a device is defined below.

a) device and DMD mutually authenticate
b) device issues mxmdp:AuthenticateReq [Figure 76].
c) DMD returns mxmdp:Ack [Figure 75].
d) Device issues mxmdp:AddDevice [Figure 81].
e) DMD adds the device identifier to the list of devices in mxmd:Device [Figure 70] part of mxmd:DomainManageInfo.
f) DMD sends a domain membership license to the device contained in the mxmdp:AddDeviceResponse message [Figure 85]
g) The device (optionally) replies with mxmdp:Ack [Figure 75].

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.

11.6.4.2 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 mxmd:User part of mxmd:DomainManageInfo. The protocol for adding a user is defined below.

a) User and DMD mutually authenticate
b) User issues mxmdp:AuthenticateReq [Figure 76].
c) DMD returns mxmdp:Ack [Figure 75].
d) User issues mxmdp:AddUser [Figure 82].
e) DMD adds the user identifier to the user list in mxmd:User [Figure 68] part of mxmd:DomainManageInfo.
f) DMD sends a domain membership license to the user contained in the mxmdp:AddDeviceResponse message [Figure 85]
g) The user (optionally) replies with mxmdp:Ack [Figure 75].

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.

11.6.4.3 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 mxmdp:AuthenticateReq [Figure 76]
c) DMD returns mxmdp:Ack [Figure 75]
d) Device issues mxmdp:RenewDevice [Figure 83]
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 mxmdp:RenewDeviceResponse message [Figure 85]
f) Else DMD applies policy and sends a mxmdp:Ack [Figure 5] with attribute "Result" set to "false" to the device
g) The device (optionally) replies with mxmdp:Ack [Figure 75].

Any Use Data recorded in the device is included in mxmdp:RenewDevice. If the DMD does not detect unlicensed simultaneous use from the Use Data (see 11.7), the DMD sends a new domain membership license containing for the device with a new expiration period.

11.6.4.4 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 mxmdp:AuthenticateReq [Figure 76].
c) DMD returns mxmdp:Ack [Figure 75]
d) User issues mxmdp:RenewUser [Figure 84]
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 mxmdp:Ack [Figure 5] with attribute "Result" set to "false" to the User.
f) The user (optionally) replies with mxmdp:Ack [Figure 75].

11.6.4.5 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 mxmdp:AuthenticateReq [Figure 76].
3. DMD returns mxmdp:Ack [Figure 75]
4. Device issues mxmdp:LeaveDevice [Figure 87]
5. DMD
    a. removes the device identifier from the mxmd:Device part of mxmd:DomainManageInfo
    b. replies with mxmdp:Ack [Figure 75] 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.

11.6.4.6 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 mxmdp:AuthenticateReq [Figure 76].
3. DMD returns mxmdp:Ack [Figure 75]
4. User issues mxmdp:LeaveUser [Figure 88].
5. DMD
    a. removes the user identifier from the list of users part of mxmd:DomainManageInfo
    b. replies with mxmdp:Ack [Figure 75] 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.

11.7 Simultaneous Content Usage Detection protocol specification

11.7.1 Introduction

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. This specification at the moment does not specify a mechanism to enforce such condition.

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

11.7.2 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 mxmd:UseData element, and contains the information that the device identified by mxmd:DeviceID used a content item belonging to m1x:isPartOf in the time frame between “Start Time” and “End Time”. It may include mxmd: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.

11.7.3 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 73], 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

11.7.4 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.

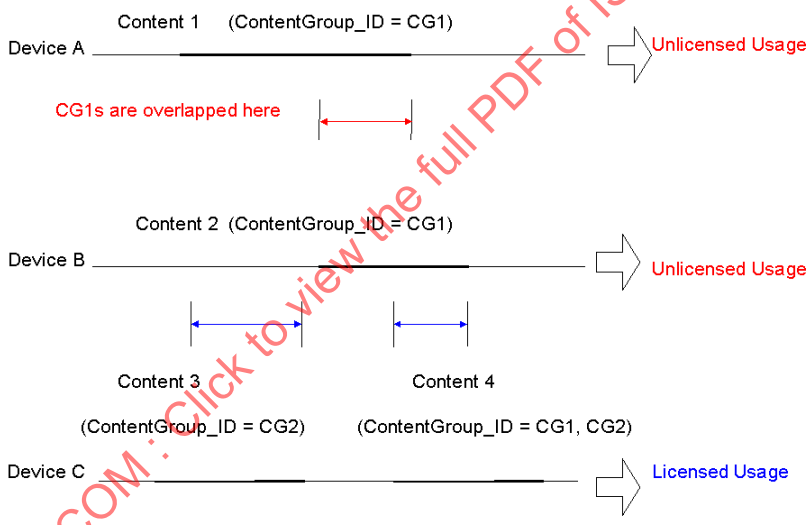

Figure 95 — Un-Licensed Simultaneous Use Schematic

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.

11.7.5 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 mxmdp:UnLicensedSimultaneousUseNotice message [Figure 93], containing all mxmdp:UseData elements where an overlapping was detected by the device. The DMD replies by sending a mxmbp:Ack [Figure 5] 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.

12 Event Reporting Protocols

12.1 Introduction

The Event Reporting protocols describe the messages exchanged between the Event Report Collection Device (ECD) and other Devices in order to implement the basic model of Event Reporting described in [11]: *“Event Reporting is required within the ISO/IEC 21000 Multimedia Framework in order to provide a standardised means for sharing information about Events amongst Peers and Users. Such Events relate to Digital Items and/or Peers that interact with them.”*

The Event Report Collection Device (ECD can be typically implemented as a server application collecting Event Reports as specified in [11]. In most scenarios, such an application has to be initialised with information regarding which Event Reports are to be registered before collecting Event Reports.

12.2 Protocols Specification

The Register Event Report Request Protocol is employed by a Device interested on being reported on the occurrence of Events which may be either directly or indirectly related either to a DI or a Device. The Event Report Request must deal on the issuing Device, or on Digital Items belonging to its User.

12.2.1 Register Event Report Request Protocol

This Protocol specifies how a Device (for example a CCD) sends this request to the ECD. This protocol is always started by the requesting Device, and it is as follows:

a) The requesting Device and the ECD mutually Authenticate.

b) The requesting Device sends to the ECD a registerERR message.

c) The ECD decides whether the requesting Device has authorisation on these messages or not.

d) The ECD replies with an mxmerp:Ack message indicating whether the request can be successfully carried out or not, specifying one of the error codes defined in msbp:BasicResultCodeType or mxmerp:ERPResultCodeType (see 8.4.2.2).

e) If the answer was OK, ECD stores the ER-R for an unlimited time or for the specified expiration time.

12.2.2 Store Event Report

The Store Event Report is employed by a Device (for example an EUD) to send an Event Report to the ECD.

This protocol is is as follows:

a) The reporting Device and the ECD mutually Authenticate.

b) The reporting Device sends to the ECD a sendER message.

c) The ECD parses the message, decides to which Devices may concern: it goes through the list of its stored ER-R reviewing to whom has to be resent.

d) The ECD sends to the requesting Device or Devices (those which had sent an ER-R) the ER.

12.3 Protocol data format

This section specifies the payload of the messages part of the Event Reporting Protocol.

12.3.1 EventReportingProtocolType

The mxmerp: EventReportingProtocolType complex type, defined in the figure below, extends the mxmbp:ProtocolType.

```
<complexType name="EventReportingProtocolType" abstract="true">
    <complexContent>
        <extension base="mxmbp:ProtocolType"/>
    </complexContent>
</complexType>
```

Figure 96 — The mxmerp:EventReportingProtocolType complex type

12.3.2 Ack

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

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

Figure 97 — The mxmerp:Ack element

The mxmerp protocol also defines some specific Event Reporting-related error codes extending the mxmbp:ExtendableResultCodeType simpleType.

```
<simpleType name="ERPResultCodeType">
    <restriction base="mxmbp:ExtendableResultCodeType">
        <enumeration value="UNAUTHORISED_ERR "/>
        <enumeration value="WRONG_PRIORITY_NUMBER"/>
        <enumeration value="WRONG_TIME_LIMITS"/>
        <enumeration value="WRONG_SPECIFICATION"/>
    </restriction>
</simpleType>
```

Figure 98 — The mxmerp:Ack element

12.3.3 RegisterERR

The mxmerp:RegisterERR message is employed to deliver one or more Event Report Requests to another Device requesting it to start collecting Event Reports related to the ERRs conveyed in the request.

```
<element name="RegisterERR" type="mxmerp:RegisterERRType"/>
<complexType name="RegisterERRType">
    <complexContent>
        <extension base="mxmerp:EventReportingProtocolType">
            <sequence>
                <element ref="erl:ERR" maxOccurs="unbounded"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 99 — The mxmerp:RegisterERR element

The RegisterERR message may contain one or more ERR and an optional Digital Signature for message integrity.

12.3.4 SendER

The mxmerp:SendER message is employed to deliver one or more Event Reports to another Device (e.g. an Event Report Collection Device) requesting it to store the Event Report(s) contained in the message.

```
<element name="SendER" type="mxmerp:SendERType"/>
<complexType name="SendERType">
    <complexContent>
        <extension base="mxmerp:EventReportingProtocolType">
            <sequence>
                <element ref="erl:ER" maxOccurs="unbounded"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Figure 100 — The mxmerp:SendER element

The SendER message may contain one or more ERs and an optional Digital Signature for message integrity.

Annex A
(informative)

Protocol Description Schemas

A.1 The MXM Base Protocol schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:mpeg:mpeg-m:schema:baseprotocol:2009"
       xmlns:mxmbp="urn:mpeg:mpeg-m:schema:baseprotocol:2009"
       xmlns="http://www.w3.org/2001/XMLSchema"
       elementFormDefault="qualified" attributeFormDefault="unqualified">
   <complexType name="ProtocolBaseType" abstract="true"/>
   <complexType name="ProtocolType" abstract="true">
      <complexContent>
         <extension base="mxmbp:ProtocolBaseType">
            <sequence>
               <element name="TransactionID" type="string"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                          Ack                               -->
   <!-- ********************************************************** -->
   <element name="Ack" type="mxmbp:AckType"/>
   <complexType name="AckType">
      <complexContent>
         <extension base="mxmbp:ProtocolType">
            <sequence minOccurs="0">
               <element ref="mxmbp:ProtocolResult"/>
            </sequence>
            <attribute name="Result" type="boolean" use="required"/>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                      ProtocolResult                        -->
   <!-- ********************************************************** -->
   <element name="ProtocolResult" type="mxmbp:ProtocolResultType"/>
   <complexType name="ProtocolResultType">
      <complexContent>
         <extension base="mxmbp:ProtocolBaseType">
            <sequence>
               <choice>
                  <element name="ResultCode" type="mxmbp:ResultCodeType"/>
                  <element name="UserDefinedResult" type="string"/>
               </choice>
               <element name="DisplayString" type="string" minOccurs="0"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <simpleType name="ResultCodeType">
      <restriction base="string">
```

```
            <enumeration value="OK"/>
            <enumeration value="PERMISSION_DENIED"/>
            <enumeration value="TIMEOUT"/>
            <enumeration value="BUSY"/>
            <enumeration value="MALFORMED_REQUEST"/>
            <enumeration value="UNABLE_TO_PROCESS"/>
            <enumeration value="OPERATION_NOT_SUPPORTED"/>
            <enumeration value="UNKNOWN_MESSAGE"/>
            <enumeration value="UNKNOWN_ERROR"/>
        </restriction>
    </simpleType>
    <!-- ********************************************************** -->
    <!--                    ContentIdentifierType                   -->
    <!-- ********************************************************** -->
    <element name="ContentIdentifier" type="mxmbp:ContentIdentifierType"/>
    <complexType name="ContentIdentifierType">
        <complexContent>
            <extension base="mxmbp:ProtocolBaseType">
                <sequence>
                    <element name="ContentItemIdentifier" type="anyURI"/>
                    <element      name="ContentElementIdentifier"      type="anyURI"
minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

Figure A.1 — The mxmbp

A.2 The MXM Access Content Protocol schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:mpeg:mpeg-m:schema:accesscontentprotocol:2009"
        xmlns:mxmacp="urn:mpeg:mpeg-m:schema:accesscontentprotocol:2009"
        xmlns="http://www.w3.org/2001/XMLSchema"
        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:rel-r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
        xmlns:rel-mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS"
        xmlns:dia="urn:mpeg:mpeg21:2003:01-DIA-NS"
        xmlns:mxmbp="urn:mpeg:mpeg-m:schema:baseprotocol:2009"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
    <import namespace="urn:mpeg:mpeg21:2002:01-DII-NS"
schemaLocation="http://www.dmpf.org/schemas/dii.xsd"/>
    <import namespace="urn:mpeg:mpeg21:2006:07-DIDL-NS"
schemaLocation="http://www.dmpf.org/schemas/didl.xsd"/>
    <import
namespace="urn:mpeg:maf:Schema:mediastreaming:IPMPINFOextensions:2007"
schemaLocation="http://www.dmpf.org/schemas/ipmpinfo-msx.xsd"/>
    <import namespace="urn:mpeg:mpeg21:2004:01-IPMPINFO-NS"
schemaLocation="http://www.dmpf.org/schemas/ipmpinfo.xsd"/>
```

```
    <import namespace="urn:mpeg:mpeg21:2003:01-REL-R-NS"
schemaLocation="http://www.dmpf.org/schemas/rel-r.xsd"/>
    <import namespace="urn:mpeg:mpeg21:2003:01-REL-MX-NS"
schemaLocation="http://www.dmpf.org/schemas/rel-mx.xsd"/>
    <import namespace="urn:mpeg:mpeg21:2005:01-REL-M1X-NS"
schemaLocation="http://www.dmpf.org/schemas/rel-m1x.xsd"/>
    <import namespace="urn:mpeg:maf:schema:mediastreaming:DIDLextensions"
schemaLocation="http://www.dmpf.org/schemas/didl-msx.xsd"/>
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-
schema.xsd"/>
    <import namespace="urn:mpeg:mpeg21:2003:01-DIA-NS"
schemaLocation="http://www.dmpf.org/schemas/ued.xsd"/>
    <import namespace="urn:mpeg:mpeg-m:schema:baseprotocol:2009"
schemaLocation="mxmbp.xsd"/>
    <!-- ################################################################ -->
    <!--                        ContentProtocolType                       -->
    <!-- ################################################################ -->
    <complexType name="AccessContentProtocolType" abstract="true">
       <complexContent>
          <extension base="mxmbp:ProtocolType"/>
       </complexContent>
    </complexType>
    <!-- ################################################################ -->
    <!--                               Ack                                -->
    <!-- ################################################################ -->
    <element name="Ack" type="mxmacp:AckType"/>
    <complexType name="AckType">
       <complexContent>
          <extension base="mxmacp:AccessContentProtocolType">
             <sequence minOccurs="0">
                <element ref="mxmbp:ProtocolResult"/>
             </sequence>
             <attribute name="Result" type="boolean" use="required"/>
          </extension>
       </complexContent>
    </complexType>
    <!-- ################################################################ -->
    <!--                          RequestContent                          -->
    <!-- ################################################################ -->
    <element name="RequestContent" type="mxmacp:RequestContentType"/>
    <complexType name="RequestContentType">
       <complexContent>
          <extension base="mxmacp:AccessContentProtocolType">
             <sequence>
                <element name="ContentIdentifier"
type="mxmbp:ContentIdentifierType"/>
                <element name="MimeType" type="string" minOccurs="0"/>
                <element ref="rel-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="mxmacp:RequestContentResponseType"/>
    <complexType name="RequestContentResponseType">
        <complexContent>
            <extension base="mxmacp:AccessContentProtocolType">
                <sequence>
                    <element name="DI" type="didl:DIDLType" minOccurs="0"/>
                    <element name="ContentURL" type="mxmacp:ContentURLType"
minOccurs="0" maxOccurs="unbounded"/>
                    <element ref="dsig:Signature" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ************************************************************** -->
    <!--                          ContentURLType                        -->
    <!-- ************************************************************** -->
    <complexType name="ContentURLType">
        <complexContent>
            <extension base="mxmbp:ProtocolBaseType">
                <sequence>
                    <element name="MimeType" type="string"/>
                    <element name="URL" type="anyURI"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

Figure A.2 — The mxmacp schema

A.3 The MXM Access IPMP Tool Protocol schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:mpeg:mpeg-m:schema:accessipmptoolprotocol:2009"
       xmlns:mxmaitp="urn:mpeg:mpeg-m:schema:accessipmptoolprotocol:2009"
       xmlns="http://www.w3.org/2001/XMLSchema"
       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:mxmbp="urn:mpeg:mpeg-m:schema:baseprotocol:2009"
       elementFormDefault="qualified" attributeFormDefault="unqualified">
    <import
namespace="urn:mpeg:maf:Schema:mediastreaming:IPMPINFOextensions:2007"
schemaLocation="http://www.dmpf.org/schemas/ipmpinfo-msx.xsd"/>
    <import namespace="urn:mpeg:mpeg21:2004:01-IPMPINFO-NS"
schemaLocation="http://www.dmpf.org/schemas/ipmpinfo.xsd"/>
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-
schema.xsd"/>
    <import namespace="urn:mpeg:mpeg-m:schema:baseprotocol:2009"
schemaLocation="mxmbp.xsd"/>
    <!-- ************************************************************** -->
    <!--                       AccessProtocolType                       -->
    <!-- ************************************************************** -->
    <complexType name="IPMPToolProtocolType" abstract="true">
        <complexContent>
```

```
            <extension base="mxmbp:ProtocolType"/>
        </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                            Ack                             -->
    <!-- ********************************************************** -->
    <element name="Ack" type="mxmaitp:AckType"/>
    <complexType name="AckType">
        <complexContent>
            <extension base="mxmaitp:IPMPToolProtocolType">
                <sequence minOccurs="0">
                    <element ref="mxmbp:ProtocolResult"/>
                </sequence>
                <attribute name="Result" type="boolean" use="required"/>
            </extension>
        </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                    RequestIPMPToolBody                     -->
    <!-- ********************************************************** -->
    <element name="RequestIPMPToolBody" type="mxmaitp:RequestIPMPToolBodyType"/>
    <complexType name="RequestIPMPToolBodyType">
        <complexContent>
            <extension base="mxmaitp:IPMPToolProtocolType">
                <sequence>
                    <element ref="ipmpinfo:IPMPToolID"/>
                    <element ref="ipmpinfo-msx:DeviceInformation"/>
                    <element ref="dsig:Signature" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                RequestIPMPToolBodyResponse                 -->
    <!-- ********************************************************** -->
    <element name="RequestIPMPToolBodyResponse"
type="mxmaitp:RequestIPMPToolBodyResponseType"/>
    <complexType name="RequestIPMPToolBodyResponseType">
        <complexContent>
            <extension base="mxmaitp:IPMPToolProtocolType">
                <sequence>
                    <choice maxOccurs="unbounded">
                        <element ref="ipmpinfo-msx:ToolBody"/>
                        <element name="ToolURL" type="anyURI"/>
                    </choice>
                    <element ref="dsig:Signature" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                RequestIPMPToolInfoList                     -->
    <!-- ********************************************************** -->
    <element                                        name="RequestIPMPToolInfoList"
type="mxmaitp:RequestIPMPToolInfoListType"/>
    <complexType name="RequestIPMPToolInfoListType">
        <complexContent>
            <extension base="mxmaitp:IPMPToolProtocolType">
                <sequence>
                    <element ref="ipmpinfo-msx:DeviceInformation" minOccurs="0"/>
```

```
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ********************************************************** -->
<!--                RequestIPMPToolInfoListResponse             -->
<!-- ********************************************************** -->
<element                                    name="RequestIPMPToolInfoListResponse"
type="mxmaitp:RequestIPMPToolInfoListResponseType"/>
<complexType name="RequestIPMPToolInfoListResponseType">
    <complexContent>
        <extension base="mxmaitp:IPMPToolProtocolType">
            <sequence>
                <element         ref="mxmaitp:IPMPToolInfo"         minOccurs="0"
maxOccurs="unbounded"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<element name="IPMPToolInfo" type="mxmaitp:IPMPToolInfoType"/>
<complexType name="IPMPToolInfoType">
    <complexContent>
        <extension base="mxmbp:ProtocolBaseType">
            <sequence>
                <element ref="ipmpinfo:IPMPToolID" />
                <element ref="ipmpmsg:ParametricDescription" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
</schema>
```

Figure A.3 — The mxmaitp schema

A.4 The MXM Access License Protocol schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:mpeg:mpeg-m:schema:accesslicenseprotocol:2009"
       xmlns:mxmalp="urn:mpeg:mpeg-m:schema:accesslicenseprotocol:2009"
       xmlns="http://www.w3.org/2001/XMLSchema"
       xmlns:didl="urn:mpeg:mpeg21:2006:07-DIDL-NS"
       xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
       xmlns:mxmbp="urn:mpeg:mpeg-m:schema:baseprotocol:2009"
       xmlns:rel-r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
       elementFormDefault="qualified" attributeFormDefault="unqualified">
    <import namespace="urn:mpeg:mpeg21:2006:07-DIDL-NS"
schemaLocation="http://www.dmpf.org/schemas/didl.xsd"/>
    <import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-
schema.xsd"/>
    <import namespace="urn:mpeg:mpeg-m:schema:baseprotocol:2009"
schemaLocation="mxmbp.xsd"/>
    <import namespace="urn:mpeg:mpeg21:2003:01-REL-R-NS"
schemaLocation="http://www.dmpf.org/schemas/rel-r.xsd"/>
    <!-- ********************************************************** -->
    <!--                   AccessLicenseProtocolType                -->
    <!-- ********************************************************** -->
```

```
    <complexType name="AccessLicenseProtocolType" abstract="true">
        <complexContent>
            <extension base="mxmbp:ProtocolType"/>
        </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                          Ack                               -->
    <!-- ********************************************************** -->
    <element name="Ack" type="mxmalp:AckType"/>
    <complexType name="AckType">
        <complexContent>
            <extension base="mxmalp:AccessLicenseProtocolType">
                <sequence minOccurs="0">
                    <element ref="mxmbp:ProtocolResult"/>
                </sequence>
                <attribute name="Result" type="boolean" use="required"/>
            </extension>
        </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                       RequestLicense                       -->
    <!-- ********************************************************** -->
    <element name="RequestLicense" type="mxmalp:RequestLicenseType"/>
    <complexType name="RequestLicenseType">
        <complexContent>
            <extension base="mxmalp:AccessLicenseProtocolType">
                <sequence>
                    <choice>
                        <element name="ContentIdentifier"
type="mxmbp:ContentIdentifierType"/>
                        <element name="LicenseID" type="anyURI"/>
                    </choice>
                    <element ref="rel-r:license" minOccurs="0"/>
                    <element ref="dsig:Signature" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ********************************************************** -->
    <!--                  RequestLicenseResponse                    -->
    <!-- ********************************************************** -->
    <element name="RequestLicenseResponse"
type="mxmalp:RequestLicenseResponseType"/>
    <complexType name="RequestLicenseResponseType">
        <complexContent>
            <extension base="mxmalp:AccessLicenseProtocolType">
                <sequence>
                    <choice>
                        <element ref="rel-r:license"/>
                        <element name="ErrorCode" type="mxmbp:ResultCodeType"/>
                    </choice>
                    <element ref="dsig:Signature" minOccurs="0"/>
                </sequence>
                <attribute name="Result" type="boolean" use="required"/>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

Figure A.4 — The mxmalp schema

A.5 The MXM Authenticate Content Protocol schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:mpeg:mpeg-m:schema:authenticatecontentprotocol:2009"
       xmlns:mxmaucp="urn:mpeg:mpeg-m:schema:authenticatecontentprotocol:2009"
       xmlns="http://www.w3.org/2001/XMLSchema"
       xmlns:mxmbp="urn:mpeg:mpeg-m:schema:baseprotocol:2009"
       xmlns:didl="urn:mpeg:mpeg21:2006:07-DIDL-NS"
       xmlns:didl-msx="urn:mpeg:maf:schema:mediastreaming:DIDLextensions"
       xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
       xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
       elementFormDefault="qualified" attributeFormDefault="unqualified">
   <import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-
schema.xsd"/>
   <import namespace="urn:mpeg:mpeg21:2006:07-DIDL-NS"
schemaLocation="http://www.dmpf.org/schemas/didl.xsd"/>
   <import namespace="urn:mpeg:mpeg-m:schema:baseprotocol:2009"
schemaLocation="mxmbp.xsd"/>
   <import namespace="urn:mpeg:maf:schema:mediastreaming:DIDLextensions"
schemaLocation="http://www.dmpf.org/schemas/didl-msx.xsd"/>
   <import namespace="urn:mpeg:mpeg21:2002:01-DII-NS"
schemaLocation="http://www.dmpf.org/schemas/dii.xsd"/>
   <!-- ************************************** -->
   <!--      AuthenticateContentProtocolType   -->
   <!-- ************************************** -->
   <complexType name="AuthenticateContentProtocolType" abstract="true">
      <complexContent>
         <extension base="mxmbp:ProtocolType"/>
      </complexContent>
   </complexType>
   <!-- ************************************** -->
   <!--             Ack                        -->
   <!-- ************************************** -->
   <element name="Ack" type="mxmaucp:AckType"/>
   <complexType name="AckType">
      <complexContent>
         <extension base="mxmaucp:AuthenticateContentProtocolType">
            <sequence minOccurs="0">
               <element ref="mxmbp:ProtocolResult"/>
            </sequence>
            <attribute name="Result" type="boolean" use="required"/>
         </extension>
      </complexContent>
   </complexType>
   <!-- ************************************** -->
   <!--      AuthenticateContentRequest        -->
   <!-- ************************************** -->
   <element name="AuthenticateContentRequest"
type="mxmaucp:AuthenticateContentRequestType"/>
   <complexType name="AuthenticateContentRequestType">
      <complexContent>
         <extension base="mxmaucp:AuthenticateContentProtocolType">
            <sequence>
               <choice>
                  <element name="DCIInfo" type="mxmaucp:InfoType"/>
                  <element name="DCI" type="didl:DIDLType"/>
               </choice>
               <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
```

```
            </extension>
        </complexContent>
    </complexType>
    <complexType name="InfoType">
        <complexContent>
            <extension base="mxmbp:ProtocolBaseType">
                <sequence>
                    <element name="ID" type="anyURI"/>
                    <element name="Signature" type="dsig:SignatureType"
minOccurs="0" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ****************************************** -->
    <!--    AuthenticateContentElementRequest      -->
    <!-- ****************************************** -->
    <element name="AuthenticateContentElementRequest"
type="mxmaucp:AuthenticateContentElementRequestType"/>
    <complexType name="AuthenticateContentElementRequestType">
        <complexContent>
            <extension base="mxmaucp:AuthenticateContentProtocolType">
                <sequence>
                    <element name="ContentElementInfo" type="mxmaucp:InfoType"/>
                    <element ref="dsig:Signature" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ****************************************** -->
    <!--          AuthenticateResponse              -->
    <!-- ****************************************** -->
    <element name="AuthenticateResponse"
type="mxmaucp:AuthenticateResponseType"/>
    <complexType name="AuthenticateResponseType">
        <complexContent>
            <extension base="mxmaucp:AuthenticateContentProtocolType">
                <sequence>
                    <choice>
                        <element name="ContentSignature" type="dsig:SignatureType"/>
                        <element name="AuthenticationResult" type="boolean"/>
                        <element name="ErrorCode" type="mxmaucp:ErrorCodeType"/>
                    </choice>
                    <element ref="dsig:Signature" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <simpleType name="ErrorCodeType">
        <restriction base="string">
            <enumeration value="HASH_MISSING"/>
            <enumeration value="HASH_CORRUPTED"/>
        </restriction>
    </simpleType>
</schema>
```

Figure A.5 — The mxmaucp schema

A.6 The MXM Domain schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:mpeg:mpeg-m:schema:domain:2009"
       xmlns:mxmd="urn:mpeg:mpeg-m:schema:domain:2009"
       xmlns="http://www.w3.org/2001/XMLSchema"
       xmlns:rel-r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
       xmlns:rel-sx="urn:mpeg:mpeg21:2003:01-REL-SX-NS"
       xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
       xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
       elementFormDefault="qualified" attributeFormDefault="unqualified">
   <import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-
schema.xsd"/>
   <import namespace="urn:mpeg:mpeg21:2003:01-REL-R-NS"
schemaLocation="http://www.dmpf.org/schemas/rel-r.xsd"/>
   <import namespace="urn:mpeg:mpeg21:2003:01-REL-SX-NS"
schemaLocation="http://www.dmpf.org/schemas/rel-sx.xsd"/>
   <import namespace="http://www.w3.org/2001/04/xmlenc#"
schemaLocation="http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-
schema.xsd"/>
   <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="mxmd:DomainIDType"/>
   <complexType name="DomainIDType">
      <complexContent>
         <extension base="mxmd:IDType">
            <sequence>
               <element ref="mxmd:DomainManagerID"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <element name="DACredentials" type="mxmd:DomainCredentialType"/>
   <element name="DomainMembershipCredentials"
type="mxmd:DomainCredentialType"/>
   <complexType name="DomainCredentialType">
      <sequence>
         <element ref="mxmd:AccessID"/>
         <element ref="mxmd:AccessPassword"/>
      </sequence>
   </complexType>
   <element name="DomainManageInfo" type="mxmd:DomainManageInfoType"/>
   <complexType name="DomainManageInfoType">
      <complexContent>
         <extension base="mxmd:DomainBaseType">
            <sequence>
               <element ref="mxmd:DomainID"/>
               <element ref="mxmd:DACredentials" minOccurs="0"/>
               <element ref="mxmd:DomainMembershipCredentials" minOccurs="0"/>
               <choice minOccurs="0" maxOccurs="2">
                  <element ref="mxmd:User"/>
                  <element ref="mxmd:Device"/>
```