INTERNATIONAL STANDARD

**ISO/IEC 23006-2**

First edition
2011-07-01

# Information technology — MPEG extensible middleware (MXM) —

## Part 2: MXM API

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

*Partie 2: API MXM*



Reference number
ISO/IEC 23006-2:2011(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 23006-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 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 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 implement media-handling value chains whose devices interoperate because they are all based on the same set of technologies.

# Information technology — MPEG extensible middleware (MXM) —

## Part 2:
## MXM API



## 1 Scope

This part of ISO/IEC 23006 specifies a set of Application Programming Interfaces (APIs) so that MXM Applications executing on an MXM Device can access the standard multimedia technologies contained in its Middleware as MXM Engines, as specified by ISO/IEC 23006-1.

The APIs belong to the following two classes:

— the MXM Engine APIs, i.e. the collection of the individual MXM Engine APIs providing access to a single MPEG technology (e.g. video coding) or to a group of MPEG technologies where this is convenient;

— the MXM Orchestrator API, i.e. the API of the special MXM Engine that is capable of creating chains of MXM engines to execute a high-level application call such as Play, as opposed to the typically low-level MXM Engine API calls.

## 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 10918-1, *Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines*

ISO/IEC 13818-2, *Information technology — Generic coding of moving pictures and associated audio information: Video*

ISO/IEC 14496-2, *Information technology — Coding of audio-visual objects — Part 2: Visual*

ISO/IEC 14496-3, *Information technology — Coding of audio-visual objects — Part 3: Audio*

ISO/IEC 14496-10, *Information technology — Coding of audio-visual objects — Part 10: Advanced Video Coding*

ISO/IEC 14496-11, *Information technology — Coding of audio-visual objects — Part 11: Scene description and application engine*

ISO/IEC 14496-16, *Information technology — Coding of audio-visual objects — Part 16: Animation Framework eXtension (AFX)*

ISO/IEC 14496-25, *Information technology — Coding of audio-visual objects — Part 25: 3D Graphics Compression Model*

ISO/IEC 15444-1, *Information technology — JPEG 2000 image coding system: Core coding system*

ISO/IEC 15948, *Information technology — Computer graphics and image processing — Portable Network Graphics (PNG): Functional specification*

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

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

ISO/IEC 21000-5, *Information technology — Multimedia framework (MPEG-21) — Part 5: Rights Expression Language*

ISO/IEC 21000-9, *Information technology — Multimedia framework (MPEG-21) — Part 9: File Format*

ISO/IEC 21000-15, *Information technology — Multimedia framework (MPEG-21) — Part 15: Event Reporting*

ISO/IEC 21000-18, *Information technology — Multimedia framework (MPEG-21) — Part 18: Digital Item Streaming*

ISO/IEC 21000-19, *Information technology — Multimedia framework (MPEG-21) — Part 19: Media Value Chain Ontology*

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

ISO/IEC 23006-4, *Information technology — MPEG extensible middleware (MXM) — Part 4: MXM protocols*

## 3 Terms and definitions

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

## 4 Abbreviated terms

| | |
|---|---|
| BBL | Bitstream Binding Language |
| DIA | Digital Item Adaptation |
| DID | Digital Item Declaration |
| DIDL | Digital Item Declaration Language |
| DII | Digital Item Identification |
| DIS | Digital Item Streaming |
| ER | Event Report |
| ERR | Event Report Request |
| IPMP | Intellectual Property Management and Protection |
| MXMD | Media Streaming Device |
| REL | Rights Expression Language |

| | |
|---|---|
| RTP | Real Time Protocol |
| RTSP | Real Time Streaming Protocol |
| 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 |
|---|---|
| ipmpdidl | urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS |
| ipmpmsg | urn:mpeg:mpeg21:2006:07-IPMPMESSAGES-NS |
| ipmpinfo | urn:mpeg:mpeg21:2004:01-IPMPINFO-NS |
| didl | urn:mpeg:mpeg21:2002:02-DIDL-NS |
| didmodel | urn:mpeg:mpeg21:2002:02-DIDMODEL-NS |
| didl-msx | urn:mpeg:maf:schema:mediastreaming:DIDLextensions |
| dii | urn:mpeg:mpeg21:2002:01-DII-NS |
| r | urn:mpeg:mpeg21:2003:01-REL-R-NS |
| sx | urn:mpeg:mpeg21:2003:01-REL-SX-NS |
| m1x | urn:mpeg:mpeg21:2005:01-REL-M1X-NS |
| xsd | http://www.w3.org/2001/XMLSchema |
| xsi | http://www.w3.org/2001/XMLSchema-instance |
| dsig | http://www.w3.org/2000/09/xmldsig# |
| 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 |

The MXM APIs are specified in the Java and C++ languages, as follows:

a) A high-level description of the interfaces defining the MXM APIs

b) An html format specification of the MXM APIs with a normative value provided as an attachment to this document

# 6 Common MXM interfaces and classes

The core part of MXM consists of a number of interfaces and classes which are common to all MXM Engines. These are:

— MXM: class enabling MXM Applications to obtain instances of MXM Engines. MXM acts as a factory instantiating those MXM engines listed in the MXM Configuration file when they are required by MXM Applications.

— MXMEngine: the interface at a highest level of an MXMEngine. Every MXMEngines extend this interface, defining a method enabling other entities to query the specific name of an MXMEngine.

— MXMObject: the basic interface for most of the MXM classes. An MXMObject is a wrapper of specific objects that can therefore be exchanged by MXM Engines. It defines the version of MXM and it provides a number of fundamental methods to know what type of object is indeed an MXMObject wrapping, what is the class name of the wrapped object, getters and setters methods for getting/setting an object from/into an MXMObject, etc.

— MXMAdapter: a basic class implementing the MXMObject interface which is a wrapper of specific objects that can therefore be exchanged by MXM Engines. By means of an MXMAdapter it is possible to convert any object into an MXMObject.

— MXMEngineName: an enumeration listing all possible names (types) of MXMEngines, so that any entity can unambiguously determine the name of an MXMEngine.

— MXMException: the highest level of exception thrown by MXM. All other exceptions in MXM extends this abstract class.

— MXMEngineResponse: an enumeration used as a general-purpose return value for a number of methods.

# 7 MXM Engine APIs

## 7.1 Digital Item Engine APIs

### 7.1.1 DIDEngine

The *DIDEngine* interface defines for operating on ISO/IEC 21000-2 Digital Item Declaration (DID) data structures. From classes implementing the DIDEngine interface it is possible to obtain instances of classes performing the main functionalities of this MXM Engine:

— classes to create Digital Items

— classes to access data contained in a Digital Item

— classes to edit a Digital Item

#### 7.1.2 DI Creation

Digital Item creation involves the following interfaces:

— DICreator: an interface defining the methods to create a Digital Item and adding a digital signature to it

— ItemCreator: an interface defining the methods to create a didl:Item after setting all its main properties

— ResourceCreator: an interface for creating a didl:Resource after setting all its main properties

#### 7.1.3 DI Editing

Digital Item editing involves the following interfaces:

— DIEditor: an interface defining the methods to edit a Digital Item

— ItemEditor: an interface defining the methods to edit a didl:Item

#### 7.1.4 DI Access

Digital Item parsing involves the following interfaces:

— DIParser: an interface defining the methods to parse a a Digital Item and retireve the information contained in it

### 7.2 MPEG-21 File Engine APIs

#### 7.2.1 MPEG21FileEngine

The MPEG21FileEngine interface defines the methods for operating over ISO/IEC 21000-9 MPEG-21 File Format files. From classes implementing the MPEG21FileEngine interface it is possible to obtain instances of:

— classes for creating an MPEG-21 file

— classes for accessing data from an MPEG-21 file

#### 7.2.2 MPEG21 File Creation

Creating an MPEG-21 file involves the following interfaces:

— MPEG21FileCreator: an interface defining the methods to create an MPEG-21 file

#### 7.2.3 MP21 File Access

Creating an MPEG-21 file involves the following interfaces:

— MPEG21FileCreator: an interface defining the methods to create an MPEG-21 file

### 7.3 REL Engine APIs

#### 7.3.1 RELEngine

The RELEngine interface defines the methods for operating over ISO/IEC 21000-5 Rights Expression Language (REL) data structures. Classes implementing the RELEngine interface act as factories creating instances of classes performing the following functionalities:

— classes to create Rights Expressions

— classes to access data contained in Rights Expression

— classes to authorise users to exercise rights

### 7.3.2 Rights Expression creation

Creating a REL statement involves the following interfaces:

— LicenseCreator: an interface defining the methods to create an r:license element

— GrantCreator: an interface defining the methods to create an r:grant

— DigitalResourceCreator: an interface defining the methods to create an r:digitalResource

— ProtectedResourceCreator: an interface defining the methods to create an m1x:protectedResource

— IdentityHolderCreator: an interface defining the methods to create an m1x:identityHolder

— IssuerCreator: an interface defining the methods to create an r:issuer

— KeyHolderCreator: an interface defining the methods to create an r:keyHolder

### 7.3.3 Rights Expression access

Parsing a REL statement involves the following interfaces:

— LicenseParser: an interface defining the methods to parse an r:license element

— GrantParser: an interface defining the methods to parse an r:grant

— DigitalResourceParser: an interface defining the methods to parse an r:digitalResource

— ProtectedResourceParser: an interface defining the methods to parse an m1x:protectedResource

— IdentityHolderParser: an interface defining the methods to parse an m1x:identityHolder

— IssuerParser: an interface defining the methods to parse an r:issuer

— KeyHolderParser: an interface defining the methods to parse an r:keyHolder

### 7.3.4 Authorisation

Authorising a user to exercise a right involves the following interfaces:

— AuthorisationManager: an interface defining the methods to authorise a user to exercise a right and retrieve information from the validation operation

— AuthorisationResult: an enumeration defining possible result of an authorisation

## 7.4 IPMP Engine APIs

### 7.4.1 IPMPEngine

The IPMPEngine interface defines the methods for operating over ISO/IEC 21000-4 Intellectual Property Management and Protection data structures. Classes implementing the IPMPEngine interface act as factories creating instances of classes performing the following functionalities:

— classes to create IPMP data structures

— classes to access data contained in IPMP data structures

### 7.4.2 IPMP Information creation

Creating IPMP data structures involves the following interfaces:

— IPMPGeneralInfoDescriptorCreator: an interface defining the methods to create an ipmpinfo:IPMPGeneralInfoDescriptor element

— IPMPInfoDescriptorCreator: an interface defining the methods to create an ipmpinfo:IPMPInfoDescriptor element

— ProtectedAssetCreator: an interface defining the methods to create an ipmpdidl:ProtectedAsset element

— RightsDescriptorCreator: an interface defining the methods to create an ipmpinfo:RightsDescriptor element

— ToolCreator: an interface defining the methods to create an ipmpinfo:Tool element

### 7.4.3 IPMP Information access

Accessing IPMP data structures involves the following interfaces:

— IPMPGeneralInfoDescriptorParser: an interface defining the methods to parse an ipmpinfo:IPMPGeneralInfoDescriptor element and retrieve information from it

— IPMPInfoDescriptorParser: an interface defining the methods to parse an ipmpinfo:IPMPInfoDescriptor element and retrieve information from it

— ProtectedAssetParser: an interface defining the methods to parse an ipmpdidl:ProtectedAsset element and retrieve information from it

— RightsDescriptorParser: an interface defining the methods to parse an ipmpinfo:RightsDescriptor element and retrieve information from it

— ToolParser: an interface defining the methods to parse an ipmpinfo:Tool element and retrieve information from it

## 7.5 Media Framework Engine APIs

The MediaFramework is a high level engine, grouping together several media specific engines such as Video, Image, Audio and Graphics Engines. It also implements common functionalities (independent on the media type) such as resource loading and saving. The following elementary media engines are currently supported: VideoEngine, AudioEngine, ImageEngine, Graphics3DEngine and each of them exposes APIs for creation (encode) and access (decode) the elementary streams.

The MediaFrameworkEngine has two interfaces:

– an interface for accessing the content (called Acces)

– an interface for creating the content (called Creation)

A typical implementation of the Acces interface of the MediaFrameworkEngine first loads a resource, demultiplex it, check the type of the elementary streams within the resource and call the associated elementary stream acces engines.

A typical implementation of the Creation interface of the MediaFrameworkEngine call the associated elementary stream creation engines, initialize them with encoding parameters and finaly save the multiplexed resource.

The table below shows which class of APIs are provided in this part of ISO/IEC 23006 for the various media categories.

**Table 2 —MXM Media Framework APIs**

| Class | Type | | Creation | Decoding | Rendering | Editing |
|---|---|---|---|---|---|---|
| Elementary Streams | Image | | Y (only JPEG) | Y | Only if it is a texture (3D) | |
| | Audio | | | | Y | |
| | Video | | | | Y | |
| Muxed content | | | Y | Y | | Y |
| Graphics-related | Scene | 2D/layout | | | Y | |
| | | 3D/scene graph | Y | Y | Y | Y |
| | Graphic primitives | 2D Graphics | | | Y | |
| | | 3D Graphics | Y | Y | Y | Y |



### 7.5.1 Access::Video APIs

Classes implementing this API are entry poins classes for decoding content defined by ISO/IEC 13818-2 (MPEG-2 Video), ISO/IEC 14496-2 (MPEG-4 Visual) or ISO/IEC 14496-10 (MPEG-4 AVC).

### 7.5.2 Access::Audio APIs

Classes implementing this API are entry poins classes for decoding content defined by ISO/IEC 14496-3 (MPEG-4 Audio).

### 7.5.3 Access::Image APIs

Classes implementing this API are entry poins classes for decoding content defined by ISO/IEC 10918-1 (JPEG), ISO/IEC 15444-1 (JPEG 2000) or ISO/IEC 15948 (PNG).

#### 7.5.4 Access::Graphics3D APIs

Classes implementing this API are entry poins classes for decoding content defined by ISO/IEC 14496-11 (Scene Description), ISO/IEC 14496-16 (Animation Framework eXtension) or ISO/IEC 14496-25 (3D Graphics Compression Model).

#### 7.5.5 Creation::Video APIs

Classes implementing this API are entry poins classes for encoding content defined by ISO/IEC 14496-2 (MPEG-4 Visual) or ISO/IEC 14496-10 (MPEG-4 AVC).

#### 7.5.6 Creation::Audio APIs

Classes implementing this API are entry poins classes for encoding content defined by ISO/IEC 14496-3 (MPEG-4 Audio).

#### 7.5.7 Creation::Image APIs

Classes implementing this API are entry poins classes for encoding content defined by ISO/IEC 10918-1 (JPEG), ISO/IEC 15444-1 (JPEG 2000) or ISO/IEC 15948 (PNG).

#### 7.5.8 Creation::Graphics3D APIs

Classes implementing this API are entry poins classes for encoding content defined by ISO/IEC 14496-11 (Scene Description), ISO/IEC 14496-16 (Animation Framework eXtension) or ISO/IEC 14496-25 (3D Graphics Compression Model).

### 7.6 Metadata Engine APIs

#### 7.6.1 MetadataEngine

The MetadataEngine interface defines the methods for operating over metadata structures. Classes implementing the MetadataEngine interface act as factories creating instances of classes performing the following functionalities:

— classes to create metadata structures, by means of the MetadataCreationEngine

— classes to access data contained in metadata structures, by means of MetadataAccessEngine

#### 7.6.2 Metadata creation

Creating metadata structures involves the following interfaces:

— MetadataCreator: a super interface to further be extended, defining a basic method to create a metadata structure

— GenericMetadataCreator: an interface defining the methods to create generic metadata structures in possibly any format depending on the specific implementation of the MetadataEngine of choice

— Mpeg7Creator: an interface defining the methods to create MPEG-7 metadata structures

— AbstractCreator: an interface defining the methods to create an mpeg7:Abstract element

— CreationCoordinatesCreator: an interface defining the methods to create an mpeg7:CreationCoordinates element

— CreationDescriptionCreator: an interface defining the methods to create an mpeg7:CreationDescription element

— CreatorCreator: an interface defining the methods to create an mpeg7:Creator element

— GenreCreator: an interface defining the methods to create an mpeg7:Genre element

— ParentalGuidanceCreator: an interface defining the methods to create an mpeg7:Creator element

— TitleMediaCreator: an interface defining the methods to create an mpeg7:Creator element

#### 7.6.3 Metadata Access

Accessing metadata structures involves the following interfaces:

— MetadataParser: a super interface to further be extended, defining a basic method to create a metadata structure

— GenericMetadataParser: an interface defining the methods to parse a generic metadata structure in possibly any format depending on the specific implementation of the MetadataEngine of choice

— Mpeg7Parser: an interface defining the methods to parse MPEG-7 metadata structures

— AbstractParser: an interface defining the methods to parse an mpeg7:Abstract element

— CreationCoordinatesParser: an interface defining the methods to parse an mpeg7:CreationCoordinates element

— CreationDescriptionParser: an interface defining the methods to parse an mpeg7:CreationDescription element

— CreatorParser: an interface defining the methods to parse an mpeg7:Creator element

— GenreParser: an interface defining the methods to parse an mpeg7:Genre element

— ParentalGuidanceParser: an interface defining the methods to parse an mpeg7:ParentalGuidance element

— TitleMediaCreator: an interface defining the methods to parse an mpeg7:TitleMedia element

### 7.7 Digital Item Streaming Engine APIs

#### 7.7.1 DISEngine

The DISEngine interface defines the methods for operating over ISO/IEC 21000-18 Digital Item Streaming data structures. Classes implementing the DISEngine interface act as factories creating instances of classes performing the following functionalities:

— classes to create Digital Item Streaming data structures and adding them into a Digital Item

— classes to stream a Digital Item

### 7.7.2 Digital Item Streaming Information creation

Creating DIS data structures involves the following interfaces:

— DISCreator: an interface defining the methods used to create Digital Item Streaming data structures and adding them into a Digital Item

### 7.7.3 Streaming of Digital Items

Streaming Digital Items involves the following interfaces:

— DISStreamer: an interface defining the methods to stream a Digital Item

## 7.8 Digital Item Adaptation Engine APIs

### 7.8.1 Introduction

The Digital Item Adaptation Engine API specifies means to access and create information pertaining to the usage environment context where Digital Items or media resources are ultimately processed, consumed, created, distributed, etc.

EXAMPLE The usage environment context may include information about the user, terminal, network, and natural environment.

### 7.8.2 Digital Item Adaptation Access

Specifies means to parse and access information contained in a usage environment description identified by namespace.

The following list provides an overview of the methods available for parsing and accessing:

- The whole usage environment description;
- Terminal capabilities;
- Encoding capabilities;
- Decoding capabilities;
- Display capabilities;
- Audio output capabilities;
- Network characteristics;
- Location;
- Time.

Further details are provided within the actual API specification.

#### 7.8.3 Digital Item Adaptation Creation

Specifies means to create information contained in a usage environment description identified by namespace.

The following list provides an overview of the methods available for parsing and accessing:

- The whole usage environment description;
- Terminal capabilities;
- Encoding capabilities;
- Decoding capabilities;
- Display capabilities;
- Audio output capabilities;
- Network characteristics;
- Location;
- Time.

Further details are provided within the actual API specification.

### 7.9 Event Reporting Engine APIs

#### 7.9.1 EREngine

The EREngine interface defines the methods for operating over ISO/IEC 21000-15 Event Reporting data structures. Classes implementing the DISEngine interface implement the methods allowing to perform the following functionalities:

- Create Event Report Requests
- Create Event Reports

### 7.10 Content Protocol Engine APIs

#### 7.10.1 ContentProtocolEngine

The ContentProtocolEngine interface defines the methods for for performing Content Protocols as specified in ISO/IEC 23006-4. Classes implementing the ContentProtocolEngine interface act as factories creating instances of classes performing the following functionalities:

- classes to Identify a content item and parts thereof from a Content Identification Device
- classes to Access content items and parts thereof from a Content Provider Device
- classes to Store a content item and parts thereof on a Content Provider Device
- classes to Authenticate a content item and/or parts thereof on a Content Provider Device

Furthermore the ContentProtocolEngine defines the methods allowing applications to register as listeners to events (i.e. callbacks) occurring once a protocol has terminated.

#### 7.10.2 Content Identification Protocol

The Content Identification Protocol involves the following interfaces:

— IdentifyContentProtocolEngine: to identify a content item or parts thereof on a remote device

— IdentifyContentProtocolEvent: an event occurring when the IdentifyContentProtocol terminates

#### 7.10.3 Content Authentication Protocol

The Content Authentication Protocol involves the following interfaces:

— AuthenticateContentProtocolEngine: to authenticate a content item or parts thereof on a remote device

— AuthenticateContentProtocolEvent: an event occurring when the AuthenticateContentProtocol terminates

#### 7.10.4 Content Store Protocol

The Content Store Protocol involves the following interfaces:

— StoreContentProtocolEngine: to store a content item or parts thereof on a remote device

— StoreContentProtocolEvent: an event occurring when the StoreContentProtocol terminates

— DCSType: interface defining the methods to store a content item for streaming

— ResourceInfoType: interface defining the methods to store a content element for streaming

#### 7.10.5 Content Access Protocol

The Content Access Protocol involves the following interfaces:

— AccessContentProtocolEngine: to access a content item or parts thereof on a remote device

— AccessContentProtocolEvent: an event occurring when the AccessContentProtocol terminates

### 7.11 License Protocol Engine APIs

#### 7.11.1 LicenseProtocolEngine

The LicenseProtocolEngine interface defines the methods for for performing License Protocols as specified in ISO/IEC 23006-4. Classes implementing the LicenseProtocolEngine interface act as factories creating instances of classes performing the following functionalities:

— classes to Access licenses from a License Provider Device

— classes to Store licenses on a License Provider Device

— classes to Revoke a license previously stored on a License Provider Device

Furthermore the LicenseProtocolEngine defines the methods allowing applications to register as listeners to events (i.e. callbacks) occurring once a protocol has terminated. Finally, the LicenseProtocolEngine defines the interfaces of the services for receiving the messages exchanged while performing the License Protocols as defined in ISO/IEC 23006-4.

#### 7.11.2 Store License Protocol

The Store License Protocol involves the following interfaces:

— StoreLicenseProtocolDispatcher: to store one or more licenses on a remote device

#### 7.11.3 Access License Protocol

The Access License Protocol involves the following interfaces:

— AccessLicenseProtocolDispatcher: to access a license from a remote device

#### 7.11.4 Revoke License Protocol

The Revoke License Protocol involves the following interfaces:

— RevokeLicenseProtocolDispatcher: to revoke a license previously stored on a remote device

### 7.12 IPMP Tool Protocol Engine APIs

#### 7.12.1 IPMPToolProtocolEngine

The IPMPToolProtocolEngine interface defines the methods for for performing IPMP Tool Protocols as specified in ISO/IEC 23006-4. Classes implementing the IPMPToolProtocolEngine interface act as factories creating instances of classes performing the following functionalities:

— classes to Access licenses from an IPMP Tool Provider Device

Furthermore the IPMPToolProtocolEngine defines the methods allowing applications to register as listeners to events (i.e. callbacks) occurring once a protocol has terminated.

#### 7.12.2 IPMP Tool Access Protocol

The Access IPMP Tool Protocol involves the following interfaces:

— AccessIPMPToolProtocolEngine: to access an IPMP Tool from a remote device

— AccessIPMPToolProtocolEvent: an event occurring when the AccessIPMPToolProtocol terminates

### 7.13 Content Search Engine APIs

#### 7.13.1 ContentSearchEngine

The ContentSearchEngine interface defines the methods for searching for content. Classes implementing the ContentSearchEngine interface implement methods providing the following functionalities:

— classes to search for content items, in a number of flavours: search by content, search by metadata and combined search

## 7.14 Security Engine APIs

### 7.14.1 SecurityEngine

The SecurityEngine interface defines security-related methods. Classes implementing the ContentSearchEngine interface implement methods providing the following functionalities:

— classes to create new credentials and manage certificates

— classes to generate symmetric keys and encrypt/decrypt data

— classes to store confidential information such as licenses and keys in the secure repository

— classes to certify the integrity of MXM tools

### 7.14.2 CertificateManager

The certificate manager involves the following interfaces:

— CertificateManager, to generate private/public keys, import and export of public keys and certificates in various formats, perform cryptographic services, secure storage and retrieval of information, generation of keys, signature calculation and validation, etc.

### 7.14.3 KeyManager

The key manager involves the following interfaces:

— KeyManager, to generate symmetric keys, providing generation of keys, hashes, signature calculation and validation, etc.

### 7.14.4 SecureDeviceManager

The Secure Device Manager involves the following interfaces:

— SecureDeviceManager, to certify and verify the integrity of MXM Devices, requesting the calculation of a fingerprint of the device with the hardware software installed, and the verification of these values.

### 7.14.5 SecureRepositoryManager

The Secure Repository Manager involves the following interfaces:

— SecureDeviceManager, to store, retrieve and manage confidential information in the secure repository.

## 7.15 MVCO Engine APIs

The MVCO Engine lets an MXM Application access the functionalities of the Media Value Chain Ontology specified in ISO/IEC 21000-19.

The Media Value Chain Ontology is is a formal representation of a set of concepts within the domain of Intellectual Property of multimedia material along the Value Chain and their relationships. The information conveyed in a Digital Item may sometimes represent Intellectual Property entities (called IP Entities). An MPEG-21 User which takes actions on Digital Items can be labelled as "author", "distributor", "consumer" etc. These terms are referred to as User Roles. The actions themselves, the actions relevant to the IP Entities are the focus of the Media Value Chain Ontology.

Through the MVCO API an MXM Application can assess the kind of IP Entity that a Digital Item represents, the role that an MPEG-21 User has played respect a certain DI etc.

The MVCOEngine APIs enable applications to perform the following functions:

— Attribute the IP Entity to a Digital Item (i.e. whether a Digital Item represents a Manifestation, an Instance or a Product).

— Cathegorise the MXM Users in roles regarding a particular IP Entity. Thus, it can be decided which MXM User is the Creator of a certain content. Note that the creator in the sense of Intellectual Property may not coincide with the creator of the Digital Item that is represented elsewhere in the metadata fields of the Digital Item.

— Track the rights owners of the represented IP Entities (the right to distribute an IP Entity may be transferred from distributors to distributors and it is interesting to know at any time who is the current rights holder.

— Trace the origin of represented IP Entities from an Intellectual Property point of view. For example, to assert which work is derived from which work etc.

— Verify that Licenses respect the Intellectual Property premises along the Value Chain.

## 7.16 Domain Engine APIs

### 7.16.1 DomainEngine

The DomainEngine interface defines methods for operating on ISO/IEC 23006-4 Domain management data structures. Classes implementing the ContentSearchEngine interface implement methods providing the following functionalities:

— classes to create and manage a domain of devices

### 7.16.2 DomainManager

The DomainManager involves the following interfaces:

— DomainManager, to create, renew and delete a domain, to add a device or a user to the domain, remove them from the domain or extend their membership to the domain; finally to request domain information.

## 7.17 Rendering Engine APIs

The rendering engine defines a number of interfaces allowing the following functionalities to be performed:

— getNumberOfScreens

— getScreen

— getAspectRatio

— getNativeResolution

— setResolution

— setPixelFormat

— setPosition

— setSize

— setEventHandler

— setSource

— renderGeometry

— showAnimation

— showBBAAnimation

# 8 MXM Orchestrator APIs

## 8.1 Digital Item Adaptation Orchestrator Engine APIs

### 8.1.1 Introduction

The Digital Item Adaptation Orchestrator Engine APIs specifies means to adapt a Digital Item or media resource based on

— the description of the content – ContentMetadataEngine;

— according to a given context – DIAEngine; and

— utilizing an actual adaptation engine – MediaFrameworkEngine.

### 8.1.2 Adapt

Specify means to adapt a Digital Item or media resource based on MetadataEngine (Subclause 7.6), DIAEngine (Subclause 7.8), and MediaFrameworkEngine (Subclause 7.5).

## 8.2 IPMP Orchestrator Engine APIs

The IPMPOrchestratorEngine APIs specifies methods to instantiate an IPMP Tool in a specific control point in the media encoding or decoding chain, initialise an instantiated IPMP Tool with initialisation data and encrypt confidential information that has to be conveyed to an IPMP Tool.

## 8.3 REL Orchestrator Engine APIs

The RELOrchestratorEngine APIs specifies high-level methods to create licenses, either generic or conforming to the Open Access Content REL profile.

## 8.4 Player Orchestrator Engine APIs

The PlayerOrchestratorEngine APIs specifies a high-level methods to play a content item of any possible format.

# Annex A
(informative)

# MXM Configuration

## A.1 Introduction

In order to enable the dynamic instantiation of MXM engines, MXM defines a schema used to create XML configuration files that specify which MXM engines shall be loaded, which properties shall they have, and other information.

The MXM Orchestrator engine is in charge of creating chains of MXM Engines and using them to execute function calls made by MXM applications. This is achieved by requesting to MXM the reference to instances of the MXM Engines which are required to satisfy the requests from MXM applications.

## A.2 The MXM Configuration file

The figure below shows an example of MXM configuration file containing some global MXM parameters and listing four MXM engines, one DIDEngine (implemented by class org.iso.mpeg.mxm.test.DIDEngine), one MPEG21FileEngine implemented by class org.iso.mpeg.mxm.test.MPEG21FileEngine and two MetadataEngines (implemented by classes org.iso.mpeg.mxm.test.MPEG7SMPMetadataEngine and org.iso.mpeg.mxm.test.GenericMetadataEngine).

It has to be noted that the "id" attribute of an MXM engine indicates the identified of a specific MXM engine. If two MXM engines of the same type, as in the case of MetadataEngine, are listed in the MXMConfiguration file, the id attribute is mandatory as it is needed to differentiate the two implementations of the same engine. The default value of the id attribute is 0, which indicates that it is the default engine of that type.

```
<?xml version="1.0" encoding="UTF-8"?>
<MXMConfiguration version="0.2.0"
          xmlns="urn:org:iso:mpeg:mxm:configuration:schema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="urn:org:iso:mpeg:mxm:configuration:schema
../../main/resources/mxmConfiguration.xsd">
    <MXMParameters>
        <!-- List here the global MXM configuration parameters, either
        a key-value one (e.g. temp folder, username, etc.) or a complex
        structure (see e.g. CustomMXMParam) -->
        <entry key="user.name">filippo</entry>
        <entry key="app.image.name">/myIcon.gif</entry>
        <ComplexParameter>
            <bar:CustomMXMParam xmlns:bar="urn:bar">A complex MXM configuration
parameter</bar:CustomMXMParam>
        </ComplexParameter>
        <MXMEnginesFolder>/usr/bin/MXMEngines</MXMEnginesFolder>
    </MXMParameters>
    <DIDEngine>
        <ClassName>org.iso.mpeg.mxm.test.DIDEngine</ClassName>
        <EngineParameters>
            <!-- List here the DIDEngine-specific configuration parameters, either
            of type key-value or a complex structure -->
            <entry key="some.didl.specific.config.param">AABBCCDD</entry>
        </EngineParameters>
```

```
        <!-- More DIDEngine-specific configuration parameters... -->

    <DIDProfileCompliance>urn:mpeg:maf:schema:mediastreaming</DIDProfileComplianc
e>
    </DIDEngine>
    <MPEG21FileEngine id="5">
        <ClassName>org.iso.mpeg.mxm.test.MPEG21FileEngine</ClassName>
        <EngineParameters>
            <!-- List here the MPEG21FileEngine-specific configuration parameters,
either
            of type key-value or a complex structure -->
            <entry key="output.folder">/Users/filippo/test</entry>
            <!-- More MPEG21FileEngine-specific configuration parameters... -->
        </EngineParameters>
    </MPEG21FileEngine>
    <MetadataEngine id="0">
        <ClassName>org.iso.mpeg.mxm.test.MPEG7SMPMetadataEngine</ClassName>
        <EngineParameters>
            <!-- More ContentMetadataEngine-specific configuration parameters -->

            <entry key="dummy.parameter">MPEG-7 test</entry>
        </EngineParameters>
    </MetadataEngine>
    <MetadataEngine id="1">
        <ClassName>org.iso.mpeg.mxm.test.GenericMetadataEngine</ClassName>
        <EngineParameters>
            <!-- More ContentMetadataEngine-specific configuration parameters -->

            <entry key="dummy.parameter">MPEG-7 test</entry>
        </EngineParameters>
    </MetadataEngine>
    <!-- All engines follow ... -->
</MXMConfiguration>
```

**Figure A.1 — An example of MXM configuration file**

## A.3 The MXM Configuration schema

The MXM configuration schema is given below.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="urn:org:iso:mpeg:mxm:configuration:schema"
        xmlns:mxm="urn:org:iso:mpeg:mxm:configuration:schema"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
        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"/>
    <complexType name="MXMConfigurationBaseType" abstract="true"/>
    <!-- ********************************************************** -->
    <!--                        MXMConfiguration                    -->
    <!-- ********************************************************** -->
    <element name="MXMConfiguration" type="mxm:MXMConfigurationType"/>
    <complexType name="MXMConfigurationType">
        <complexContent>
            <extension base="mxm:MXMConfigurationBaseType">
                <sequence>
```

```
                <element name="MXMParameters" type="mxm:MXMParameterType"
minOccurs="0"/>
                <element ref="mxm:MXMEngine" minOccurs="0"
maxOccurs="unbounded"/>
                <element ref="dsig:Signature" minOccurs="0"/>
            </sequence>
            <attribute name="version" type="string" use="required"/>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="MXMParameterType">
      <complexContent>
         <extension base="mxm:MXMGenericParameterType">
            <sequence>
               <element name="MXMEnginesFolder" type="string" minOccurs="0"
maxOccurs="unbounded"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="MXMGenericParameterType">
      <complexContent>
         <extension base="mxm:MXMConfigurationBaseType">
            <sequence>
               <element name="entry" type="mxm:KeyValueParameterType"
minOccurs="0" maxOccurs="unbounded"/>
               <element name="ComplexParameter" type="mxm:ComplexParameterType"
minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="KeyValueParameterType">
      <simpleContent>
         <extension base="string">
            <attribute name="key" type="string" use="required"/>
         </extension>
      </simpleContent>
   </complexType>
   <complexType name="ComplexParameterType">
      <complexContent>
         <extension base="mxm:MXMConfigurationBaseType">
            <sequence>
               <any namespace="##other" processContents="lax"
maxOccurs="unbounded"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                          Engines                           -->
   <!-- ********************************************************** -->
   <element name="MXMEngine" type="mxm:MXMEngineType" abstract="true"/>
   <complexType name="MXMEngineType" abstract="true">
      <complexContent>
         <extension base="mxm:MXMConfigurationBaseType">
            <sequence>
               <element name="DynamicLibrary" type="mxm:DynamicLibraryType"
minOccurs="0"/>
               <element name="ClassName" type="string"/>
```

```
                <element name="EngineParameters"
type="mxm:MXMGenericParameterType" minOccurs="0"/>
            </sequence>
            <attribute name="id" type="string" default="0" use="optional"/>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="DynamicLibraryType">
      <complexContent>
         <extension base="mxm:MXMConfigurationBaseType">
            <sequence>
               <element name="LibraryPath" type="string"/>
               <element name="LoadMode" type="string" minOccurs="0"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                          DIEngine                          -->
   <!-- ********************************************************** -->
   <element name="DIDEngine" type="mxm:DIDEngineType"
substitutionGroup="mxm:MXMEngine"/>
   <complexType name="DIDEngineType">
      <complexContent>
         <extension base="mxm:MXMEngineType">
            <sequence>
               <element name="DIDProfileCompliance" type="anyURI"/>
               <!-- More DIDEngine-specific configuration parameters TBD -->
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                       MPEG21FileEngine                     -->
   <!-- ********************************************************** -->
   <element name="MPEG21FileEngine" type="mxm:MPEG21FileEngineType"
substitutionGroup="mxm:MXMEngine"/>
   <complexType name="MPEG21FileEngineType">
      <complexContent>
         <extension base="mxm:MXMEngineType">
            <sequence>
               <!-- MPEG21FileEngine-specific configuration parameters TBD -->
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ********************************************************** -->
   <!--                          RELEngine                         -->
   <!-- ********************************************************** -->
   <element name="RELEngine" type="mxm:RELEngineType"
substitutionGroup="mxm:MXMEngine"/>
   <complexType name="RELEngineType">
      <complexContent>
         <extension base="mxm:MXMEngineType">
            <sequence>
               <!-- RELEngine-specific configuration parameters TBD -->
            </sequence>
         </extension>
      </complexContent>
   </complexType>
```