INTERNATIONAL STANDARD

ISO/IEC 23006-3

First edition
2011-02-01

Information technology — MPEG extensible middleware (MXM) —

Part 3: MXM reference software

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

Partie 3: Logiciel de référence MXM

Reference number
ISO/IEC 23006-3:2011(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 23006-3 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 hosting MXM applications thanks to the existence of a standard MXM architecture,

— MXM components thanks to the existence of standard MXM components APIs,

— 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 3:
MXM reference software

1 Scope

This part of ISO/IEC 23006 describes the reference software implementing the normative clauses of ISO/IEC 23006-1. The information provided is applicable for determining the reference software modules available for ISO/IEC 23006-1, understanding the functionality of the available reference software modules, and utilizing the available reference software modules.

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

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

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

IPMP Intellectual Property Management and Protection

LPD License Provider Device

REL Rights Expression Language

URI Uniform Resource Identifier

5 MXM software overview

5.1 The MXM software repository

The figure below shows a graphical representation of the MXM software.

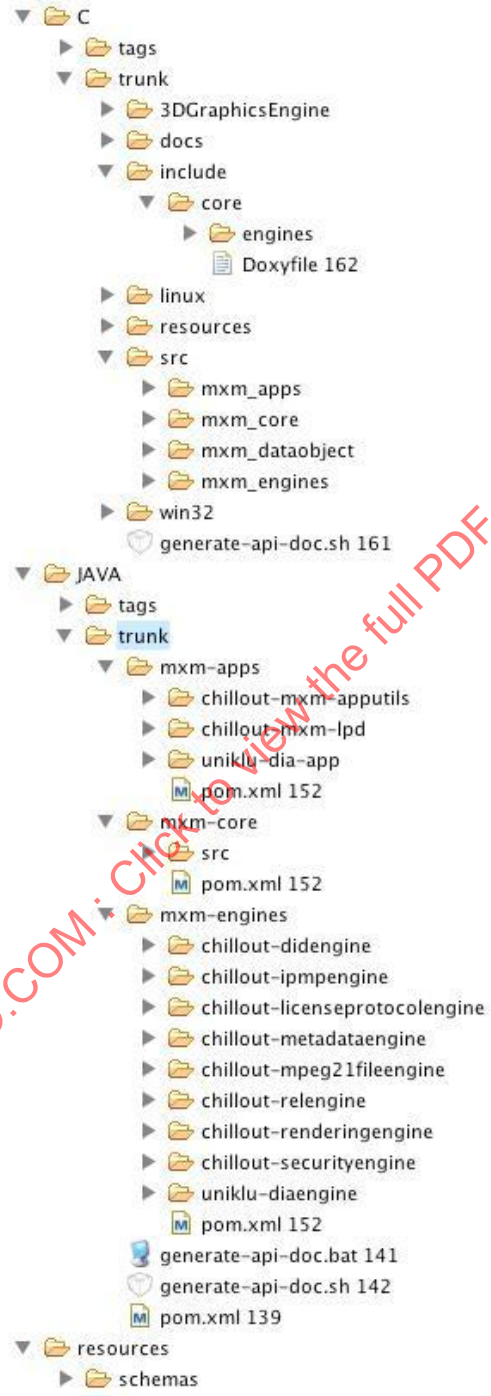

Figure 1 — The MPEG-M software structure

NOTE The MPEG-M remote folder contains two main sub-folders: C and Java, containing the MXM API definition:

— In C++: C/include/core/engines/

— In Java: JAVA/mxm-core/src/main/java

The reference software also contains preliminary MXM software implementations in both programming languages, as well as all the schemas referred in ISO/IEC 23006-1, ISO/IEC 23006-2 and ISO/IEC 23006-4.

6 MXM Java software implementation

6.1 Introduction

The Java MXM software is available in source code under the terms of the BSD License [4]. The software is organised in a number of projects, whose build is managed by Maven [7]. All projects have a common structure:

- src/main/java --> Containing the main classes and/or interfaces organised in packages
- src/main/resources --> Containing the resources (e.g. xml files, properties files, etc.) used by the engine or the application at runtime
- src/test/java --> Containing the unit test classes used to test the main classes
- src/test/resources --> Containing those resources which are used by the tests when being executed.

The Java MXM software is divided in the following modules:

- **mxm-core**: the core MXM project. It defines the MXM APIs and it provides a thin layer of code implementing MXM as a factory of MXM Engines based on the contents of the MXMConfiguration file passed over to MXM when instantiated. Summarising, mxm-core contains the following categories of classes:
  - all interfaces defining the MXM API
  - a number of classes for:
    - parsing the MXM configuration file (conforming to the MXM configuration schema) defining which MXM Engines shall be loaded.
    - loading MXM Engines. MXM Core acts as a factory providing MXM applications one or more instances of the MXM Engines listed in the MXM configuration file.
- **mxm-engines**: containing the software implementation of MXM engines. Currently a preliminary version of the following MXM Engines has been developed:
  - DIDEngine
  - DIAEngine
  - MPEG21FileEngine
  - MetadataEngine
  - RELEngine
  - IPMPEngine
  - SecurityEngine
  - LicenseProtocolEngine
  - ContentProtocolEngine
  - DISEngine
  - EREngine

— **mxm-applications**: containing examples of MXM applications which can be customised or used for conformance test purposes. The following MXM applications have been provided:

  — chillout-mxm-lpd

  — uniklu-dia-app

6.2 Java MXM Engines

6.2.1 The DIDEngine

A preliminary implementation of the DID Engine has been provided by the Chillout project [10]. The DIDEngine is used to generate and conversely to parse a Digital Item. By means of this engine it is possible to add to or retrieve from a Digital Item the following Content and Content element information:

— identifiers

— related identifiers

— item IDs

— MPEG-7 (MDS) metadata

— IPMPGeneralInfoDescriptors

— protectedAssets

— laserObjects

— signatures

— resource information (mimeType, ref, encoding)

6.2.2 The DIA Engine

This Subclause describes a portion of the MXM reference software related to `adaptResByDescr`, `setUseEnvDescr`, and `parseUseEnvDescr` as defined in ISO/IEC 21000-7.

The software is structured as follows:

— `org.iso.mpeg.mxm` provides general classes for the MPEG MXM API.

— `org.iso.mpeg.mxm.demo` provides a simple demo application utilizing the actual reference software.

— `org.iso.mpeg.mxm.dia` provides the MXM API related to MPEG-21 Digital Item Adaptation.

— `org.iso.mpeg.mxm.exception` provides the MXM API Exceptions.

The software assumes that an `MXMResource` implements the `MXMResAdaptDescrI` which defines the `adaptResByDescr`. That is, the `MXMResource` [1] is adapted according to the Usage Environment Description (UED) which is parsed using `MXMDIAUED`. `MXMDIAUED` implements both `setUseEnvDescr` and `parseUseEnvDescr` methods.

[1] The `MXMResource` contains an URI that identifies the actual media resource to be adapted.

NOTE `parseUseEnvDescr` has been modified and now returns an `Object` instead of a `String`. It is recommended to change this in the WD1.0 of MXM APIs accordingly.

In the following, the main steps of the `adaptResByDescr` method are described:

1) Read/parse the `UsageEnvironmentType` from the `String:ued` using the DIA reference software via the MXM API.

2) Create a new `MXMDIAEngine` and adapt the resource.

The MPEG-21 DIA reference software is used to parse the UED into its internal representation and is then available as `DIA` object. The actual adaptation extracts the relevant information from the DIA `UsageEnvironmentProperty` objects (e.g., codec capabilities, display capabilities, etc.) and triggers the adaptation process based on these parameters. The adaptation is performed using the `mencoder` which is executed through Java's `Runtime.exec()` environment. Thus, the execution of this reference software requires a proper installation of the `mencoder` including `libavcodec`.

For further documentation – specifically, caveats and obstructions of the `MXMDIAEngine` – the interested reader is referred to the Java documentation.

6.2.3 The MPEG21FileEngine

The Chillout MPEG21 File Engine is used to generate and conversely to parse ISO/IEC 21000-9 (MPEG-21 File Format) files. By means of this engine it is possible to create or to extract from an MPEG-21 file the following information:

— A Digital Item (to/from the XML Box)

— A number of resources (to/from the MediaData Box)

6.2.4 The MetadataEngine

A preliminary implementation of the Metadata Engine has been provided by the Chillout project [10]. The Chillout ContentMetadata Engine is used to generate and conversely to parse MPEG-7 (MDS) metadata structures. By means of this engine it is possible to add to or retrieve from an MPEG-7 description the following Content and Content element information:

— CreationDescription containing
  — titles
  — titleMedia
  — abstracts
  — creators
  — creationCoordinates
  — copyrightStrings
  — genres
  — parentalGuidances

— contentManagementDescriptions

— classificationSchemeDescriptions

— contentEntityDescriptions

6.2.5 The RELEngine

A preliminary implementation of the REL Engine has been provided by the Chillout project [10]. The Chillout REL Engine is used to generate and conversely to parse ISO/IEC 21000-5 data structures. By means of this engine it is possible to add to or retrieve from an REL license the following information:

— Grants containing

  — Principals

  — Right

  — Resource

  — Conditions

— Issuer

This Engine also supports the creation as well as the retrieval of other information from a License as MXMObjects, thus allowing to manage more complex ISO/IEC 21000-5 data structures.

The inventory is a collection of LicensePart elements. Each LicensePart is identified by a lincensePartId attribute that must be set. In order to use a LicensePart element created inside the Inventory, it's enough to declare an empty element having licensePartIdRef attribute set. Actually the LicensePart that can be used are: DigitalResource, KeyHolder, ProtectedResource, IdentityHolder, ServiceReference, DiReference and other elements like Conditions.

The second important part in the license is the Grant. The Grant is made up of the following elements: ForAll, Principal, Right, Resource, Conditions.

1. ForAll: is the element that allow to store information that can be useful hereinafter. Each variable is defined by a ForAll element with a varName attribute set. Actually the only element type that can be used with the ForAll is the PropertyPossessor element. The PropertyPossessor element lets declare a set of TrustRoot members of the same URI. Using the variables stored in the ForAll elements is very similar to using Inventory elements, but in this case it's enough to declare an empty element having the varRef attribute set.

2. Principal: this element represent the recipient of the Grant. The elements that can be used to declare a Principal are two: KeyHolder and IdentityHolder. Another way to represent a Principal is using the variable stored in the ForAll elements. Using this expedient, it is possible to declare a Pricipal like a PropertyPossessor, so allowing to set more than one Principal for a Resource.

3. Right: this element expresses what kind of “action” can be done with a Grant. For instance if the Right is “play” then the Principal will be able to play the Resource.

4. Resource: with this element the Resource (or Service) that the Grant permits to use can be represented. Actually the Resource that can be used are: DigitalResource, ProtectedResource, Grant, GrantGroup, ServiceReference, DiReference. As soon as possible all the other Resource will be implemented. Using the Grant/GrantGroup resource it is possible to create a chains of rights.

5. Conditions: this element explains how and when the Resource can be consumed; if nesessary it’s possible to specify whether a fee per use should be paid.

The last element in the license is the Issuer. This element represent the entity that has issued the license.

Finally the Chillout REL Engine is capable of performing a basic license authorisation procedure, evaluating a query against a license and providing applications with the result.

6.2.6 The IPMPEngine

A preliminary implementation of the IPMP Engine has been provided by the Chillout project [10]. The Chillout IPMP Engine is used to generate and conversely to parse ISO/IEC 21000-4 and ISO/IEC 23001-3 data structures. By means of this engine it is possible to add to create as well as parse the following information:

— IPMPInfoDescriptors and IPMPGeneralInfoDescriptors

— (IPMP)ToolDescriptions

— InitializationSettings and InitialisationData

— ProtectedAssets

— Various IPMP XML messages

— etc.

6.2.7 The SecurityEngine

A preliminary example implementation of the IPMP Engine has been provided by the Chillout project [10]. The Chillout Security Engine is used to generate and conversely to parse Digital Signature and XML Encryption data structures, to create, store and manage digital certificates and symmetric/asymmetric keys, and to store sensitive data in a secure repository.

The MXM Security Engine provides a number of security-related functionalities that, when operating combined with external hardware or software tools, may enable applications to operate in a security-aware context.

6.2.8 The LicenseProtocolEngine

A preliminary implementation of the License Protocol Engine has been provided by the Chillout project [10]. The Chillout License Protocol Engine is used to generate and dispatch the messages defined in the MXM License Protocol (part of ISO/IEC 23006-4), in both a synchronous and asynchronous fashion,. Moreover, the LicenseProtocolEngine defines the interfaces of the services that applications (i.e. License Provider Devices) wishing to implement the server part of the License Protocols may implement for ease of implementation.

6.2.9 The ContentProtocolEngine

A preliminary implementation of the Content Protocol Engine has been provided by the Chillout project [10]. The Chillout Content Protocol Engine is used to generate and dispatch the messages defined in the MXM Content Protocol (part of ISO/IEC 23006-4), in both a synchronous and asynchronous fashion. Moreover, the ContentProtocolEngine defines the interfaces of the services that applications (i.e. Content Provider Devices) wishing to implement the server part of the Content Protocols may implement for ease of implementation. By means of this engine is now possible to:

— Identify a Content

— Access a Content

— Store a Content

6.2.10 The DISEngine

A preliminary implementation of the DIS Engine has been provided by the Chillout project [10]. The DISEngine is used to generate and conversely to parse a BBL file. This file contains information about time-dependent metadata packets. A metadata packet is represented by an XML document where are specified an image, a text annontation, a start time and a duration. The can be added to a DI as further metadata. By means of this engine it is possible:

— Add time-dependent metadata packets to a DI

— Generate the corresponding BBL file

— Retrieve information about the packets to be streamed together with the DI

6.2.11 The EREngine

The EREngine is used to generate and conversely parse an Event Report Request. By means of this engine it is possible to add to or retrieve from an Event Report Request the following information:

— Request Descriptor

— Request Specification

— Payload Specification

— Digital Item Metadata

— Delivery Specification

— Transport Service

— Request Condition Descriptor

This is only a preliminary implementation that generates static ERRs following the protocol directives.

6.3 Java MXM Applications

6.3.1 The Chillout MXM LPD

The Chillout MXM License Provider Device is an example MXM Application running in the Apache Tomcat servlet container which can be used as a starting point to develop a License Provider Device as well as a test bed for testing applications implementing the client side of the License Protocols.

This example application uses Apache Axis to expose the License Provider Device services which can be invoked over the SOAP protocol to request or store licenses. In order to run the Chillout MXM LPD all is required is to deploy the chillout-mxm-lpd.war file which is generated when building the application. Its services can be tested by running the JUNIT test classes in package org.dmp.chillout.mxm.licenseprotocolengine.test available as part of the LicenseProtocolEngine MXM Engine.

7 MXM C++ software implementation

7.1 MediaFramework Engine

7.1.1 MediaFrameworkAcces Engine

The MediaFrameworkAcces is a set of libraries implementing the interface MXMFileFormat.

The following table provides the name and the functionality of each library

Library Name	Library Functionalities
libMP4Decoder, MP4ClassesLib	MP4 demux libraries

7.1.1.1 Access::Graphics3D Engine

The Acces::Graphics3D is a set of libraries implementing the following interfaces MXMGeometry, MXMApparence, MXMTexture and MXMAnimation.

The following table provides the name and the functionality of each library

Library Name	Library Functionalities
3DMCLib	Decoder of the MPEG-4 AFX-3DMC Elementary Stream
BBADecPC	Decoder of the MPEG-4 AFX-BBA Elementary Stream
BIFStoVBLib	Library ensuring the conversion between IndexedFaceSet node and a flat representation
Gpac	Decoder of MPEG-4 BIFS
Jasper-1.701.0	Decoder of JPEG 2000 Elementary Stream
libFAMC	Decoder of MPEG-4 AFX FAMC Elementary Stream
MP4toVBLib	Utilities libraries

7.1.2 MediaFrameworkCreation Engine

The MediaFrameworkCreation is a set of libraries implementing the interface MXMFileFormat.

The following table provides the name and the functionality of each library

Library Name	Library Functionalities
libMP4Encoder, MP4ClassesLib	MP4 mux libraries

7.1.2.1 Creation::Graphics3D Engine

The Creation::Graphics3D is a set of libraries implementing the following interfaces MXMGeometry, MXMApparence, MXMTexture and MXMAnimation.

The following table provides the name and the functionality of each library

Library Name	Library Functionalities
3DMCLib	Encoder of the MPEG-4 AFX-3DMC Elementary Stream
BBAEncoderLib	Encoder of the MPEG-4 AFX-BBA Elementary Stream
Gpac	Encoder of MPEG-4 BIFS
Jasper-1.701.0	Encoder of JPEG 2000 Elementary Stream
libFAMC	Encoder of MPEG-4 AFX FAMC Elementary Stream
VBtoMP4Lib	Utilities libraries

Annex A
(informative)

Check out the MXM source code from the MXM svn repository

In order to checkout the source code you need a subversion [11] client.

Open the terminal window, cd to the folder that will contain the MXM source code and type the following command:

Java: svn co http://wg11.sc29.org/mxmsvn/repos/JAVA/trunk mxm

C++: svn co http://wg11.sc29.org/mxmsvn/repos/C/trunk mxm-cpp

The following user has read access to the repository:

- mxmpubro with mpegmxmro password