INTERNATIONAL STANDARD

ISO/IEC 23006-3

Third edition
2016-12-01

Information technology — Multimedia service platform technologies —

Part 3: **Conformance and reference software**

Technologies de l'information — Technologies de la plate-forme de services multimédia —

Partie 3: Conformité et logiciel de référence

Reference number
ISO/IEC 23006-3:2016(E)

Contents

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This third edition cancels and replaces the second edition (ISO/IEC 23006-3:2013), which has been technically revised.

A list of all parts in the ISO/IEC 23006 series can be found on the ISO website.

Introduction

The ISO/IEC 23006 series 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, especially MPEG technologies, accessible from the middleware APIs, elementary services and aggregated services.

The ISO/IEC 23006 series is referred to as MPEG Extensible Middleware (MXM) in its first edition, and it specifies an architecture (ISO/IEC 23006-1), an API (ISO/IEC 23006-2), a conformance and reference software (ISO/IEC 23006-3) and a set of protocols which MXM Devices had to adhere (ISO/IEC 23006-4). It specifies also how to combine elementary services into aggregated services (ISO/IEC 23006-5).

The ISO/IEC 23006 series is subdivided in five parts:

Part 1 — Architecture: specifies the architecture that can be used as a guide to an MPEG-M implementation;

Part 2 — MPEG Extensible Middleware (MXM) Application Programming Interface (APIs): specifies the middleware APIs;

Part 3 — Conformance and Reference Software (the present document): specifies conformance criteria and a reference software implementation with a normative value;

Part 4 — Elementary Services: specifies elementary service protocols between MPEG-M applications;

Part 5 — Service Aggregation: specifies mechanisms enabling the combination of Elementary Services and other services to build Aggregated Services.

Information technology — Multimedia service platform technologies —

Part 3: Conformance and reference software

1 Scope

This document describes the reference software implementing the normative clauses of ISO/IEC 23006-1, ISO/IEC 23006-2 and ISO/IEC 23006-4 and specifies conformance criteria. The information provided are 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.

The conformance profiles are applicable to MPEG-M Services as defined in ISO/IEC 23006-4 and in ISO/IEC 23006-5.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 23006-2:2016, *Information technology — MPEG-M (Multimedia Service Platform Technologies) — Part 2: MPEG extensible middleware (MXM) API*

ISO/IEC 23006-4:2013, *Information technology — MPEG-M (Multimedia Service Platform Technologies) — Part 4: Elementary services*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

3.1.1
Aggregated Service
service resulting from the combination of *Elementary Services* (3.1.2)

3.1.2
Elementary Service
basic unit of *service* (3.1.13)

3.1.3
content
Digital Item and its component elements, namely resources (e.g. media, scripts, executable), identifiers, descriptions (e.g. metadata), event reports

3.1.4
contract
set of metadata, *license* (3.1.8), promises and signers agreed by users of a multimedia value chain, where a promise is a signed collection of statements about, e.g. obligations, prohibitions and assertions, and a signer is a *user* (3.1.15) whose signature makes the contract valid

3.1.5
device
hardware/software or simply software apparatus that enables a *user* (3.1.15) to play a role in multimedia *value chains* (3.1.16)

3.1.6
event
performance of a specified set of functions or operations

3.1.7
entity
one of the following elements in the multimedia value chain: *content* (3.1.3), *contract* (3.1.4), *device* (3.1.5), *event* (3.1.6), *license* (3.1.8), *service* (3.1.13), and *user* (3.1.15)

3.1.8
license
collection of authorizations, conditions and payment terms granted by a *user* (3.1.15) to other users

3.1.9
protocol
set of rules and data format used by two *devices* (3.1.5) to communicate

3.1.10
resource
individually identifiable asset or a sequence of assets such as a video or audio clip, a 3D synthetic scene, an image, a textual asset, a 2D LASeR scene, a web page, a single program or a full 24 h programming of a TV broadcast, a script or executable, etc.

3.1.11
right
ability of a *user* (3.1.15) to perform an operation in the multimedia *value chain* (3.1.16)

3.1.12
role
ability of a *user* (3.1.15) to perform a set of operations in the multimedia *value chain* (3.1.16)

3.1.13
service
operation performed on an *entity* (3.1.7) by a *user* (3.1.15) on behalf of other users

3.1.14
service provider
user (3.1.15) offering *services* (3.1.13) to other users

3.1.15
user
participant in multimedia *value chains* (3.1.16)

3.1.16
value chain
collection of users , including creators, end users and service providers, that conform to this document

3.1.17
MPEG-M Application
application that runs on an *MPEG-M Device* (3.1.18) and makes calls to the MPEG-M Application API and *MPEG-M Engine* (3.1.19) APIs

3.1.18
MPEG-M Device
device (3.1.5) equipped with a selected set of *MPEG-M Engines* (3.1.19)

3.1.19
MPEG-M Engine
collection of specific technologies that are bundled together to provide a specific functionality that is needed by *MPEG-M Applications* (3.1.17)

3.1.20
MPEG-M Engine API
API of a single *MPEG-M Engine* (3.1.19)

3.1.21
MPEG-M Orchestrator API
API of the *MPEG-M Orchestrator Engine* (3.1.22).

3.1.22
MPEG-M Orchestrator Engine
special *MPEG-M Engine* (3.1.19) capable of creating chains of *MPEG-M Engines* (3.1.19)

EXAMPLE To set-up a sequence of connected *MPEG-M engines* (3.1.19) for the purpose of executing a high-level application call, such as Play.

3.1.23
MPEG-M Technology
technology that is required to implement (a profile of) MPEG-M

3.2 Abbreviated terms

AIT	Advanced IPTV Terminal
AS	Aggregated Service
BBL	Bitstream Binding Language
BPMN	Business Process Model and Notation
CEL	Contract Expression Language
DI	Digital Item
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

ES	Elementary Service
IPMP	Intellectual Property Management and Protection
IPTV	Internet Protocol Television
MDS	Multimedia Description Schemes
MPEG	Moving Picture Experts Group
MPEG-21	Multimedia Framework [see ISO/IEC 21000 (all parts)]
MPEG-A	Multimedia Application Format [see ISO/IEC 23000 (all parts)]
MPEG-M	Multimedia Service Platform Technologies [see ISO/IEC 23006 (all parts)]
MPEG-V	Multimedia Context and Control [see ISO/IEC 23005 (all parts)]
MPQF	MPEG Query Format
REL	Rights Expression Language
RTP	Real Time Protocol
RTSP	Real Time Streaming Protocol
SE	Sensory Effect
SEM	Sensory Effect Metadata
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

4 Namespaces and conventions

For clarity, throughout this document, consistent namespace prefixes are used.

"`xml:`" and "`xmlns:`" are normative prefixes defined in W3C XMLNAMES. The prefix "`xml:`" is by definition bound to "http://www.w3.org/XML/1998/namespace". The prefix "`xmlns:`" is used only for namespace bindings and is not itself bound to any namespace name.

"`xsi:`" prefix is not normative. It is a naming convention in this document to refer to an element of the http://www.w3.org/2001/XMLSchema-instance namespace. All other prefixes used in either the text or examples of this specification are not normative, e.g. "`mpegm:`", "`dia:`".

In particular, most of the informative examples in this document are provided as XML fragments without the normally required XML document declaration and, thus, miss a correct namespace binding context declaration.

Unless specified otherwise, all unqualified descriptions fragments assume the default namespace "`urn:mpeg:mpegM:schema:02-service-NS:2012`".

In these descriptions fragments, the different prefixes are bound to the namespaces as given in Table 1. The schema locations of the namespaces in Table 1 are only an informative indication as schema locations may change over time.

Table 1 — Mapping of prefixes to namespaces used in examples and text

Prefix	Corresponding namespace	Schema location
mpegm	urn:mpeg:mpegM:schema:02-service-NS:2011	
mpegmb	urn:mpeg:mpegM:schema:01-base-NS:2011	
dia	urn:mpeg:mpeg21:2003:01-DIA-NS	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/dia-2nd/UED-2nd.xsd
erl	urn:mpeg:mpeg21:2005:01-ERL-NS	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/er/er.xsd
fru	urn:mpeg:mpegB:schema:FragmentRequestUnits:2007	Defined in ISO/IEC 23001-2:2008
mpeg7	urn:mpeg:mpeg7:schema:2004	
mpeg7s	urn:mpeg:mpeg7:systems:2001	
cel	urn:mpeg:mpeg21:cel:contract:2011	
bbl	urn:mpeg:mpeg21:2007:01-BBL-NS	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/dis/bbl.xsd
dii	urn:mpeg:mpeg21:2002:01-DII-NS	
mpqf	urn:mpeg:mpqf:schema:2008	Defined in ISO/IEC 15938-12
mpeg4ipmp	urn:mpeg:mpeg4:IPMPSchema:2002	Defined in ISO/IEC 14496-13:2004
ipmpdidl	urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/ipmp/ipmp-didl.xsd
ipmpmsg	urn:mpeg:mpegB:schema:IPMP-XML-MESSAGES:2007	Defined in ISO/IEC 23001-3:2008
ipmpinfo	urn:mpeg:mpeg21:2004:01-IPMPINFO-NS	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/ipmp/ipmp-general.xsd
didl	urn:mpeg:mpeg21:2002:02-DIDL-NS	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/did/didl.xsd
mpegm-didl	urn:mpeg:mpegM:schema:06-didl-NS:2012	

Table 1 *(continued)*

Prefix	Corresponding namespace	Schema location
didmodel	urn:mpeg:mpeg21:2002:02-DIDMODEL-NS	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/did/didmodel.xsd
didl-msx	urn:mpeg:maf:schema:mediastreaming:DIDLextensions	Defined in ISO/IEC 23000-5:2011
dii	urn:mpeg:mpeg21:2002:01-DII-NS	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/dii/dii.xsd
rel-r	urn:mpeg:mpeg21:2003:01-REL-R-NS	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/rel-r/rel-r.xsd
rel-sx	urn:mpeg:mpeg21:2003:01-REL-SX-NS	http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-21_schema_files/rel-r/rel-sx.xsd
xsd	http://www.w3.org/2001/XMLSchema	http://www.w3.org/2001/XMLSchema.xsd
xsi	http://www.w3.org/2001/XMLSchema-instance	
dsig	http://www.w3.org/2000/09/xmldsig#	http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd
xenc	http://www.w3.org/2001/04/xmlenc#	
bpmn	http://www.omg.org/spec/BPMN/20100524/MODEL	http://www.omg.org/spec/BPMN/20100501/BPMN20.xsd
bpmnext1	urn:mpeg:mpegM:schema:04-bpmn-ext-mfr-NS:2012	
ca	urn:mpeg:mpegM:service-type:04-content-adaptation-NS:2012	
cel	urn:mpeg:mpeg21:2010:01-CEL-NS	
cidl	urn:mpeg:mpeg-v:2010:01-CIDL-NS	
dc	http://purl.org/dc/elements/1.1/	http://dublincore.org/schemas/xmls/qdc/2008/02/11/dc.xsd
dcdv	urn:mpeg:mpeg-v:2010:01-DCDV-NS	
ebucore	urn:ebu:metadata-schema:ebuCore_2010	http://www.ebu.ch/metadata/schemas/EBUCore/20100820/EBU_CORE_20100820.xsd

Table 1 *(continued)*

Prefix	Corresponding namespace	Schema location
esi	urn:mpeg:mpegM:service-type:03-extract-sensory-information-NS:2012	
etsi	urn:dvb:metadata:iptv:sdns:2008-1	Defined in ETSI TS 102 034[22]
ipmpinfo-msx	urn:mpeg:maf:Schema:mediastreaming:IPMPINFOextensions:2007	Defined in ISO/IEC 23000-5:2011
rs	urn:mpeg:mpegM:service-type:01-recognize-speech-NS:2012	
saml	urn:oasis:names:tc:SAML:2.0:assertion	http://docs.oasis-open.org/security/saml/v2.0/saml-schema-assertion-2.0.xsd
samlp	urn:oasis:names:tc:SAML:2.0:protocol	http://docs.oasis-open.org/security/saml/v2.0/saml-schema-protocol-2.0.xsd
sedl	urn:mpeg:mpeg-v:2010:01-SEDL-NS	
sepv	urn:mpeg:mpeg-v:2010:01-SEPV-NS	
sev	urn:mpeg:mpeg-v:2010:01-SEV-NS	
sid	urn:mpeg:mpegM:schema:05-sid-NS:2012	
ss	urn:mpeg:mpegM:service-type:02-synthesize-speech-NS:2012	
tva	urn:tva:metadata:2010	Defined in ETSI TS 102 822-3-1[23]
wsdl	http://www.w3.org/ns/wsdl	http://www.w3.org/2002/ws/desc/ns/wsdl20.xsd
xsl	http://www.w3.org/1999/XSL/Transform	http://www.w3.org/2007/schema-for-xslt20.xsd
xhtml	http://www.w3.org/1999/xhtml	http://www.w3.org/MarkUp/SCHEMA/xhtml11.xsd

5 Reference software overview

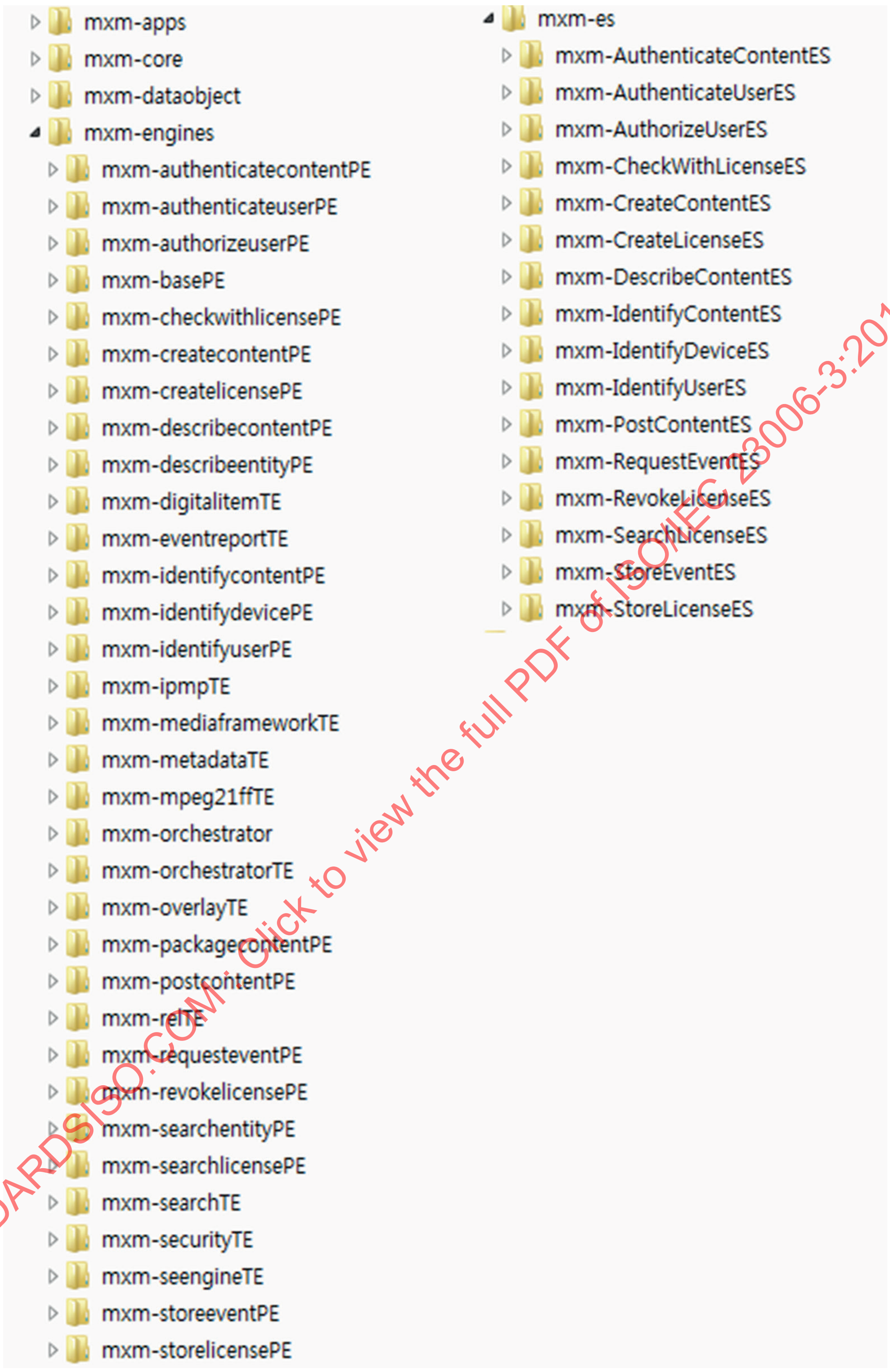
5.1 General

The reference software described in this document covers all the main functionalities highlighted in ISO/IEC 23006-1, ISO/IEC 23006-2 and ISO/IEC 23006-4.

5.2 The MXM software repository

Figure 1 shows a graphical representation of the MXM software structure. The MXM software is provided in attach or on the official MPEG-M svn repository.[17]

The JAVA trunk folder contains all the JAVA MXM reference software modules. At the moment no C/C++ reference software for MXM is provided.



```
▷ mxm-apps
▷ mxm-core
▷ mxm-dataobject
◢ mxm-engines
    ▷ mxm-authenticatecontentPE
    ▷ mxm-authenticateuserPE
    ▷ mxm-authorizeuserPE
    ▷ mxm-basePE
    ▷ mxm-checkwithlicensePE
    ▷ mxm-createcontentPE
    ▷ mxm-createlicensePE
    ▷ mxm-describecontentPE
    ▷ mxm-describeentityPE
    ▷ mxm-digitalitemTE
    ▷ mxm-eventreportTE
    ▷ mxm-identifycontentPE
    ▷ mxm-identifydevicePE
    ▷ mxm-identifyuserPE
    ▷ mxm-ipmpTE
    ▷ mxm-mediaframeworkTE
    ▷ mxm-metadataTE
    ▷ mxm-mpeg21ffTE
    ▷ mxm-orchestrator
    ▷ mxm-orchestratorTE
    ▷ mxm-overlayTE
    ▷ mxm-packagecontentPE
    ▷ mxm-postcontentPE
    ▷ mxm-relTE
    ▷ mxm-requesteventPE
    ▷ mxm-revokelicensePE
    ▷ mxm-searchentityPE
    ▷ mxm-searchlicensePE
    ▷ mxm-searchTE
    ▷ mxm-securityTE
    ▷ mxm-seengineTE
    ▷ mxm-storeeventPE
    ▷ mxm-storelicensePE
◢ mxm-es
    ▷ mxm-AuthenticateContentES
    ▷ mxm-AuthenticateUserES
    ▷ mxm-AuthorizeUserES
    ▷ mxm-CheckWithLicenseES
    ▷ mxm-CreateContentES
    ▷ mxm-CreateLicenseES
    ▷ mxm-DescribeContentES
    ▷ mxm-IdentifyContentES
    ▷ mxm-IdentifyDeviceES
    ▷ mxm-IdentifyUserES
    ▷ mxm-PostContentES
    ▷ mxm-RequestEventES
    ▷ mxm-RevokeLicenseES
    ▷ mxm-SearchLicenseES
    ▷ mxm-StoreEventES
    ▷ mxm-StoreLicenseES
```

Figure 1 — MXM software structure

6 MXM Java software implementation

6.1 General

The Java MXM software is available in source code under the terms of the BSD License.[18] The software is organized in a number of projects, whose build is managed by Maven.[21] 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 mxm-core, mxm-engines, mxm-es, mxm-applications and mxm-dataobject modules.

6.2 mxm-core (normative)

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 MXM Configuration File passed to MXM when instantiated. Summarizing, mxm-core contains the following categories of classes:

— all the interfaces defining the MXM API conforming but not limited to ISO/IEC 23006-2 API specifications of Protocol and Technology Engines;

— a number of classes for:

  — parsing the MXM Configuration File (conforming to the MXM Configuration Schema depicted in ISO/IEC 23006-2:2016, Annex A) defining which MXM Engines specific implementations shall be loaded;

  — loading the specified MXM Engines. mxm-core acts as a factory providing to the MXM Applications one or more instances of the MXM Engines listed in the MXM Configuration File.

The mxm-core is the part of MXM that essentially enables the interoperability between the various implementations of MXM Engines and guarantees the conformance of the software to the standard. The former is achieved by:

— using a general container to carry the data units;

— defining all the interfaces provided by the MXM APIs, so that, even if their implementation is different, the same calls will be made to the Technology Engines transparently, using the methods defined in the core interfaces.

The latter is achieved because the MXM Engines implement a standard interfaces; hence, an MXM Application that only relies on the standard APIs to make the calls to the MXM Technology Engines will only work if the underlying middleware implementation conforms to the standard APIs.

MXM defines a fundamental interface called MXMObject and the class implementing is the MXMAdapter. These two are the basic containers for the data exchanged between the MXM Engines ensuring the interoperability between different MXM Engine implementations. Every information unit has to be wrapped inside an MXMObject to be passed to another MXM Engine, which in turn converts it back to a data structure that can be handled.

Other important parts of mxm-core are the MXM Engine related classes and interfaces that define the basic operations that all MXM Engines should implement, their possible responses and a generic exception class. Furthermore, it defines the generic interfaces that should be implemented by the classes handling the portions of data in the engines (i.e. classes creating and parsing the data units exchanged by the engines).

The mxm-core is responsible for initializing the MXM Engines, for checking if they exist and they are operational and for exposing them to the MXM Orchestrators and other higher level components. The mxm-core discovers the MXM Engines using the MXM Configuration File, which provides a mapping between the engine name and the JAVA classes that implements the MXM Engine APIs in each middleware instance.

Finally, the mxm-core defines a generic event class called MXMEvent. This provides a level of flexibility to the technology engines implementation, in the sense that they can asynchronously communicate, without having any dependencies between them.

6.3 mxm-engines (informative)

It contains the current software implementations of MXM Engines APIs. The list of MXM Engines already developed is:

- Technology Engines:
  - Digital Item Engine;
  - MPEG-21 File Format Engine;
  - REL Engine;
  - IPMP Engine;
  - Media Framework Engine;
  - Metadata Engine;
  - Event Reporting Engine;
  - Security Engine;
  - Search Engine;
  - Contract Engine;
  - Overlay Engine;
  - Sensory Effect Engine;
- Protocol Engines:
  - Authenticate Content and User Engines;
  - Authorize User Engine;
  - Check with License Engine;
  - Create Content and License Engines;
  - Describe Content Engines;
  - Identify Content, Device and User Engines;
  - Package Content Engines;
  - Post Content Engines;
  - Request Event Engines;

— Revoke License Engines;

— Search License Engine;

— Store Event and License Engine.

This several components contain the implementation of the MXM Engines APIs for reference software. The MXM Engines implement the corresponding standard interfaces that have been defined in the mxm-core module (and specified in ISO/IEC 23006-2). The MXM Engines shall be implemented in such a way that they can exist independently of each other.

6.4 mxm-es (informative)

It hosts the interface definitions of the elementary services defined in ISO/IEC 23006-4. Bearing in mind that current state-of-the-art frameworks which can be exploited to implement MPEG-M reference software (Web Services, RMI, etc.) are using such interfaces to define what the client and the server part should implement, this package provide the single point of reference for both the server and the client implementation.

6.5 mxm-applications (informative)

It contains examples of MXM Applications which can be customised or used for conformance test purposes. The following MXM applications have been provided:

— DIMaker: it permits to create a Digital Item locally via DigitalItemTE or remotely via CreateContentPE, identify it remotely via IdentifyContentPE, embedding in it a Metadata description and save it locally in an MP21 File;

— DIReader: it permits to access a Digital Item retrieved from a MPEG21 File, retrieve from it the Resource URL and pass it to the MediaFrameworkTE to render on display;

— DISignerVerifier: it permits to sign a Digital Item and to verify if the Digital Item in the meantime was tampered or not;

— DMAGTests: it permits to read an example's license and to call the Authorize User ES using information from the license and the passed parameters (user, right and resource). Moreover it permits to parse the same license and to make calls to Revoke License Service, Store License Service and Search License Service, invoking local implementations;

— LicenseMaker: it permits to create remotely a dummy License to embed in a Digital Item;

— User Authenticator: it permits to check remotely if a User is authenticated to do some kind of operations.

6.6 mxm-dataobject (informative)

It contains the data representation of the XML schemas managed by the MXM Engines. These schemas may either be related to the data structures that the MXM technology Engines are called to deal with (for example, the REL Engine has to be able to deal with the REL XML schema, specified in ISO/IEC 21000-5) or with the protocols specified for the elementary services and implemented by the Protocol Engines.

The reference MXM software implementation relies on the Java Architecture for XML Binding (JAXB)[26] to bind the XML schemas to Java classes that can, then, be easily handled by the MXM Engines.

The way to generate from sketch the JAXB classes is highlighted in Annex A.

6.7 Java MXM Technology Engines

6.7.1 Digital item engine

A preliminary implementation of the Digital Item Engine has been provided by the Chillout project[24] and has been refactored by the Convergence project.[28] The Digital Item Engine 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:

— Content Identifier;

— Related content identifiers;

— Inner Item Identifiers;

— MPEG-7 metadata;

— IPMPGeneralInfoDescriptors and ProtectedAssets;

— LaserObjects;

— Signatures;

— Resource information (mimeType, ref, encoding).

6.7.2 MPEG21 File Format Engine

The Chillout[24] MPEG21 File Engine and its extension made by the Convergence project[28] are 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 and 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).

6.7.3 REL Engine

A preliminary implementation of the REL Engine has been provided by the Chillout project[24] and has been extended by the Convergence project.[28] The 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 REL License as implementations of mxm-core interfaces, thus allowing managing more complex ISO/IEC 21000-5 data structures.

The inventory of a REL license is a collection of LicensePart elements. Each LicensePart is identified by a licensePartId attribute that shall be set. In order to use a LicensePart element created inside the Inventory, it is enough to declare an empty element having licensePartIdRef attribute set. Actually, the

LicensePart that can be only 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: is the element that allows 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 is enough to declare an empty element having the varRef attribute set.

— Principal: this element represents 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 Principal like a PropertyPossessor, so allowing to set more than one Principal for a Resource.

— 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 if the Conditions are met.

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

— Conditions: this element explains how and when the Resource can be consumed; if necessary, it is possible to specify whether a fee per use should be paid.

The last element in the license is the Issuer. This element represents the entity that issues the license.

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

6.7.4 IPMP Engine

A preliminary implementation of the IPMP Engine has been provided by the Chillout project.[24] 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, IPMPGeneralInfoDescriptors and ToolDescriptions;

— InitializationSettings and InitialisationData;

— ProtectedAssets;

— IPMP XML messages.

6.7.5 Media Framework Engine

The implementation of the Media Framework Engine, provided by CEDEO, is based on the GStreamer libraries.[20] The Media Framework Engine can be used to render a passed resource on a specified java.awt.Canvas. The Canvas can also be omitted; in this case a java.swing.JFrame will be created with a java.awt.Canvas embedded inside.

To be able to use this Media Framework Engine within an MPEG-M Application, final users have to install the GStreamer libraries (for Windows, available on http://code.google.com/p/ossbuild/downloads/list) and launch the MPEG-M Application passing the path where the GStreamer libraries are installed, i.e. -Djna.library.path="<your_path_to_gstreamer>\v0.10.6\lib". Moreover, to enable logging of the GStreamer libraries, please set the GStreamer environment logging variable: “GST_DEBUG” to the value “*:3” or higher.

The GStreamer Media Framework implements the Access Media side of the Media Framework APIs depicted in ISO/IEC 23006-3, but can be easily adapted to implement also the Creation Media side.

6.7.6 Metadata Engine

A preliminary implementation of the Metadata Engine has been provided by the Chillout project.[24] This implementation has been updated by the Convergence Project.[28] The Metadata 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.

6.7.7 Event Report Engine

The Event Report engine is used to generate and conversely parse an Event Report and Event Report Request. The current implementation provides the following functionalities:

— creation of ERRs and ERs respectively, through dedicated APIs (Creator classes) for the description of their components;

— storage and retrieval of the ERs and ERRs respectively, through dedicated APIs (Keeper classes). The storage is done locally and the ER and ERRs can be retrieved either directly (using ids) or though subscription VDI identifiers;

— access to ERRs and ERs respectively, through dedicated APIs (Parser classes) for accessing their components.

The Event Report engine runs a service on a particular device and listens to incoming messages on a dedicated Service Access Point and it is orchestrated through the OrchestratorTE to handle newly created ERs.

6.7.8 Security Engine

The 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 Security Engine also includes methods which provide atomic actions needed for challenge-response authentication.

The 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.

6.7.9 Search Engine

The implementation of the Search Engine that has been provided by the Convergence project[28] is able to create and parse MPQF queries. By means of this engine, it is possible to add to or retrieve from a MPQF Query the following:

— InputQuery;

— QueryConditions including the following:

  — logical conditions : AND/OR/NOT;

  — comparison conditions : Equal/NotEqual/LessThan/GreaterThan;

  — query type conditions : QueryBySPARQL /QueryByFreeText.

6.7.10 Contract Engine

The implementation of the Contract Engine is still in progress.

6.7.11 Overlay Engine

The implementation of the Overlay Engine has been provided by the Convergence project[28] and is able to provide the following functionality:

— maintain an overlay of MPEG-M Devices. This overlay may be a distributed one (meaning that a distributed overlay of MPEG-M Devices is implemented) or a centralized one. In the former case, the devices are exchanging content between themselves, while in the latter case they are using a centralized service to send this content;

— send Content to the other MPEG-M Devices of the overlay or to the centralized service hosting them.

6.7.12 Sensory Effect Engine

The Sensory Effect Engine is used to manipulate, retrieve, adapt and actuate SEM structures. The current implementation provides the following functionalities:

— accessing SEM structures respectively, through dedicated APIs(SE manipulation classes) for manipulating SEMs;

— retrieving SEM structures respectively, through dedicated APIs(SE retrieval classes) for searching SEMs;

— getting the information of device capabilities, user preferences, sensor capabilities and sensed data respectively, through dedicated APIs(SE adaptation classes) for mapping SEMs;

— setting device commands of the actuator respectively, through dedicated API(SE actuation class) for actuating device commands.

6.8 Java MXM Protocol Engines

The Protocol Engines, whose API is specified in ISO/IEC 23006-2, are handling the protocols specified by the corresponding Elementary Services. Their implementation shall not affect the functionality of the Elementary Services, as specified in ISO/IEC 23006-4, as the latter is performed only by the technology engines that can be orchestrated by the Orchestrator Engine in the context of an Elementary Service.

Given that, the Elementary Services functionality is actually performed by the corresponding Orchestrator Engine, as specified in ISO/IEC 23006-2, the higher layers (e.g. applications) shall be able to access it either locally or remotely, using a common API. As the entry point for the Elementary Services is the Protocol Engines, they shall implement both a local and a remote call to the right orchestration. For this reason, MXM provides two implementations: the first one makes a local call to the orchestration, while the second one makes a remote call to the service making the call to the orchestration. More information on the exact implementation of the Elementary Services is provided below.

The local part of the Protocol Engines API is implemented to make a request to the corresponding Orchestrator Engine, which is defined in the MXM Configuration File supplied by the application. The remote part of the Protocol Engines API is implemented to make a web service call to the Elementary Service. The parameters required to make the service call are included in the MXM Configuration File.

6.8.1 Java MXM Elementary Services

The MXM Elementary Services are implemented as web services, without this being restrictive, in the sense that there can be different implementations, as long as they are staying in line with the protocol specification of the ISO/IEC 23006-4. The Elementary Service skeleton part is essentially an application running on the server side, where the middleware contains the needed set of Technology Engines to perform the specified functionality.

Hence, following the same guidelines as the MXM Applications, the Elementary Services shall be provided with an MXM Configuration File, that specifies the Engines needed to execute their functionality. The minimum set of these engines will contain the schema handler of the corresponding Protocol Engine, as this is required to parse the request and form the response defined by the protocol. Moreover, the Orchestrator Engine will often be provided, as ISO/IEC 23006-2 specifies one Orchestrator Engine for each Elementary Service, which orchestrates the Technology Engines needed to implement this Elementary Service functionality.

6.8.2 Technical guidelines

The implementation of the Protocol Engines and the Elementary Services is tightly connected, since the former are essentially performing the operations implied by the definition of the latter in ISO/IEC 23006-4. An elementary service is only responsible for handling the communication between the client and the server side. The actual implementation of the protocol is in the hands of the Protocol Engine. This means that we can have two versions of the Protocol Engine implementation:

— The Protocol Engine relies on a chain of Technology Engines, set up and called by the corresponding orchestrator. These engines are responsible for carrying out the actual protocol operation.

— The Protocol Engine makes a remote call using the corresponding Elementary Service. On the remote side, the skeleton of the Elementary Service makes a call to its local (server) implementation of the protocol, which takes to the first step. After it has been completed, the result is returned back to the Elementary Service client (stub) which delivers it to the local (client) implementation of the protocol engine.

In either of the two aforementioned cases, the Protocol Engine is implemented by extending and implementing the abstract engine definition that lies in mxm-core module. Hence, the call to the Protocol Engine is transparent to the application layer, it only depends on the MXM Configuration File to point to the first or the second choice.

Furthermore, another package is specified in the mxm-core module, named “es”, that hosts the interface definitions of the Elementary Services. Bearing in mind that current state-of-the-art frameworks which can be exploited to implement MPEG-M reference software (Web Services, RMI, etc.) are using such interfaces to define what the client and the server part should implement, this package provided in the mxm-core is used as single entry point of reference for both the server and the client implementation.

The type of the input/output parameters of the Elementary Service interface is out of the scope of MPEG-M; the developers can decide what best suits their needs. So, for example, a developer may choose to serialize all requests/responses xml documents in strings and, then, send them as such over the network. On the other hand, others may depend on implementations such as JAX that are supposed to dynamically perform the necessary marshalling/unmarshalling of the given schemas, which allows the direct use of the schema classes in an Elementary Service remote interface.

6.8.3 Create Content Usage scenario

In Figure 2, an example of the schema used for the Create Content PE and ES is depicted, as implemented for the purposes of Convergence project.[28] In detail, the abstract definition of the Create Content Engine is included in the mxm-core project. Then, two implementations of the Create Content Protocol Engine are created: the first in the package named createcontentPE.local and the second one in createcontentPE.remote. While the former is the actual implementation of the engine (see Ver.1 in Figure 2), the latter is the Elementary Service client (see Ver.2 in Figure 2).

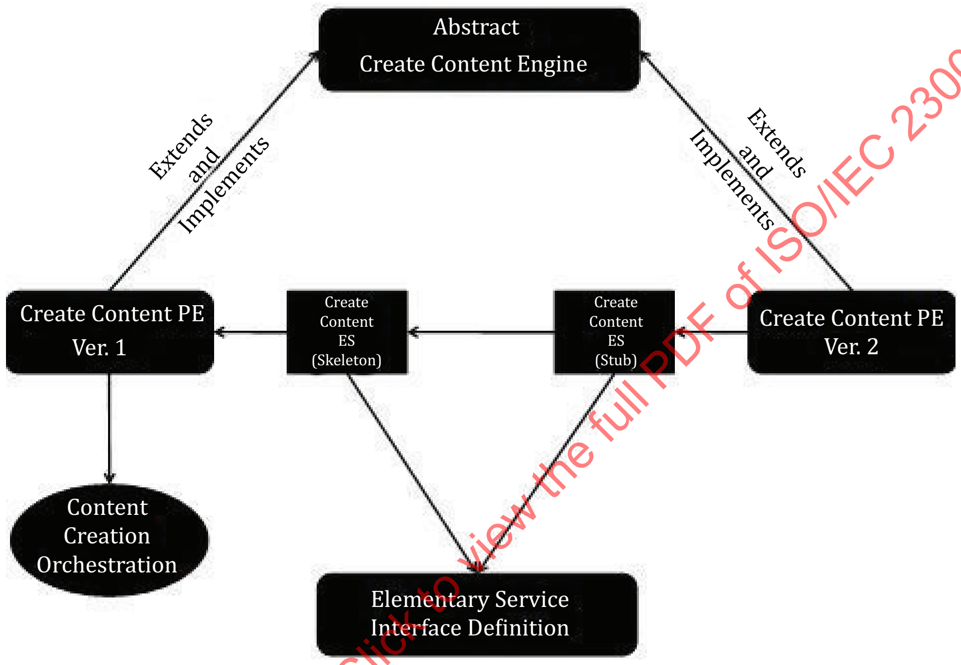


Figure 2 — Create Content example

In case an MPEG-M Device holds the first version, then it does not need any Elementary Service to contact, since it can perform locally the call and satisfy the application request. On the other hand, for the second case, the MPEG-M Device running the application needs to make a remote call to the elementary service, which is hosted on another MPEG-M device, such as the one mentioned for the first case.

The Elementary Service is defined through a simple interface, in the module mxm-core, in the package org.iso.mpeg.mxm.es.createcontentES and it is used for implementing both the server and the client side. In current implementation, JAX-WS 2.0 framework is used.

7 Profiles

7.1 Overview

This document specifies two profiles for MSPT. MPEG-M Services may support either of these profiles.

7.2 “strict” profile

The profile “strict” specifies that the following metadata formats shall be supported:

— Content metadata shall be externally represented as MPEG-7 MDS. An example of Content metadata is shown in ISO/IEC 23006-4:2013, F.1.1.

— Device metadata shall be externally represented as MPEG-21 DIA UED. An example of Device metadata is shown in ISO/IEC 23006-4:2013, F.2.1.

— Service Instance descriptions shall be externally represented as Service Instance Declaration specified in Annex E. An example of Service Instance Declaration is shown in ISO/IEC 23006-4:2013, F.3.

— User metadata shall be externally represented as MPEG-21 DIA UED with Social Metadata extension to MPEG-7 MDS specified in ISO/IEC 23006-4:2013, Annex G. An example of User metadata is shown in ISO/IEC 23006-4:2013, F.4.1.

7.3 “lax” profile

The profile “lax” specifies that metadata about Content, Device, Service, and User can be externally represented in any metadata format. For example, Content metadata can be represented in TV-Anytime format and EBU Core format as shown in ISO/IEC 23006-4:2013, F.1.2 and F.1.3, respectively.

The metadata formats supported by the Service Provider are indicated in the Service Instance Declaration.

7.4 ProfileCS

The profiles are listed in the ProfileCS Classification Scheme. This Classification Scheme is used by the Service Instance Declaration in ISO/IEC 23006-4, to indicate the Profile that the Service Provider supports.

```
<?xml version="1.0" encoding="UTF-8"?>
<mpeg7:Mpeg7 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
xsi:schemaLocation="urn:mpeg:mpeg7:schema:2004
http://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-7
_schema_files/mpeg7-v2.xsd">
  <mpeg7:Description xsi:type="mpeg7:ClassificationSchemeDescriptionType">
    <!-- ****************************************************** -->
    <!--              ProfileCS                                  -->
    <!-- ****************************************************** -->
    <mpeg7:ClassificationScheme uri="urn:mpeg:mpegM:cs:04-pro-
file-NS:2011:ProfileCS" domain="//ServiceInstanceDeclaration/Profile">
      <mpeg7:Header xsi:type="mpeg7:DescriptionMetadataType">
        <mpeg7:Version>1.0</mpeg7:Version>
      </mpeg7:Header>
      <mpeg7:Term termID="1">
        <mpeg7:Name xml:lang="en">Strict</mpeg7:Name>
      </mpeg7:Term>
      <mpeg7:Term termID="2">
        <mpeg7:Name xml:lang="en">Lax</mpeg7:Name>
      </mpeg7:Term>
    </mpeg7:ClassificationScheme>
  </mpeg7:Description>
</mpeg7:Mpeg7>
```

Annex A
(informative)

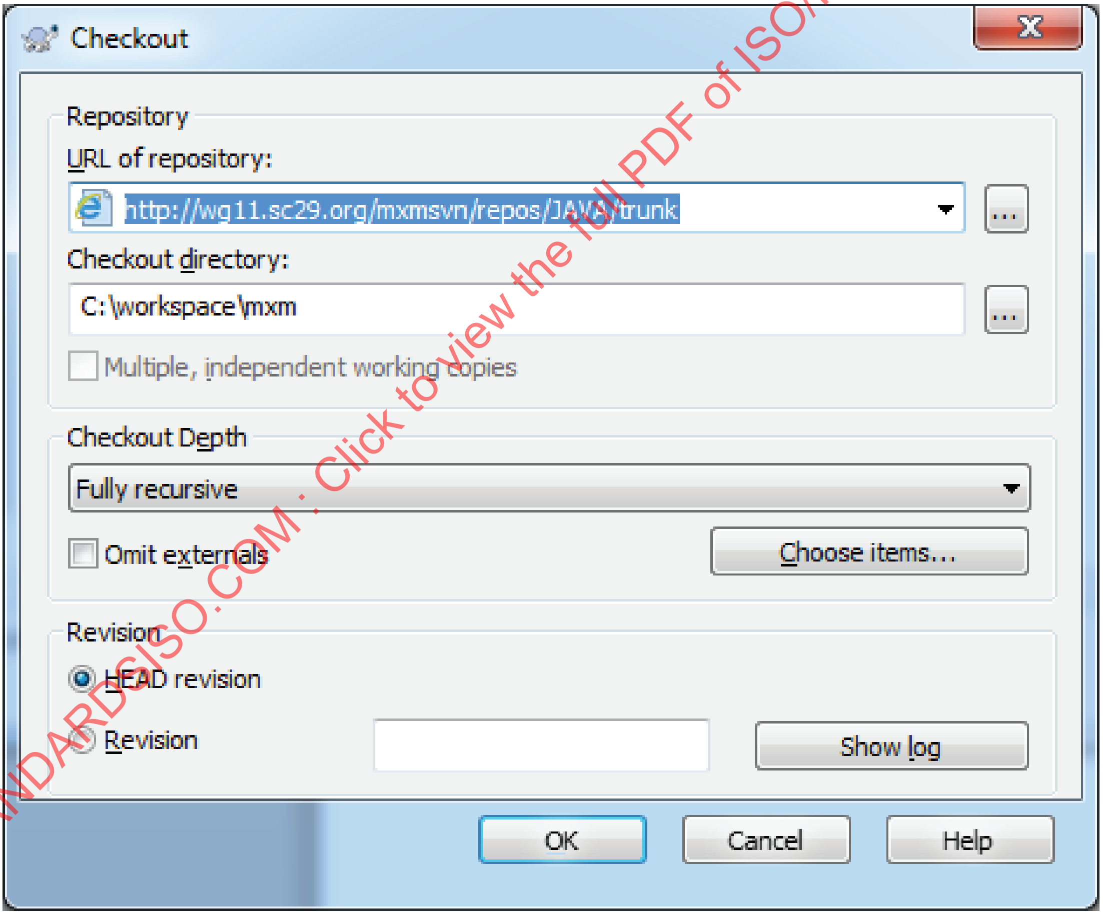
Check out of MXM source code from the MXM svn repository

In order to check out the source code you need a subversion[25] client or tortoiseSVN.[29]

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

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

Figure A.1 shows how to download MXM source code by using tortoiseSVN in the Windows environment. The URL of repository is http://wg11.sc29.org/mxmsvn/repos/JAVA/trunk, set Checkout directory as the directory that will contain the MXM source code and then press OK.


Checkout

Repository

URL of repository: http://wg11.sc29.org/mxmsvn/repos/JAVA/trunk

Checkout directory: C:\workspace\mxm

[ ] Multiple, independent working copies

Checkout Depth

Fully recursive

[ ] Omit externals

Revision

[✓ HEAD revision] Revision

Figure A.1 — Downloading MXM source code using tortoiseSVN in the Windows environment

The repository is open for read access with Username: “mxmpubro” and Password: “mpegmxmro”. For write access, please contact the MPEG-M mailing list.[27]

Annex B
(informative)

Building of MXM JAVA reference software

B.1 Software requirements

In order to build the MXM Java software, the following softwares are required:

— Java: http://www.oracle.com/technetwork/java/javase/downloads/

— Eclipse: http://www.eclipse.org/downloads/

— Maven: http://maven.apache.org/download.cgi

— Doxygen: http://www.stack.nl/~dimitri/doxygen/download.html

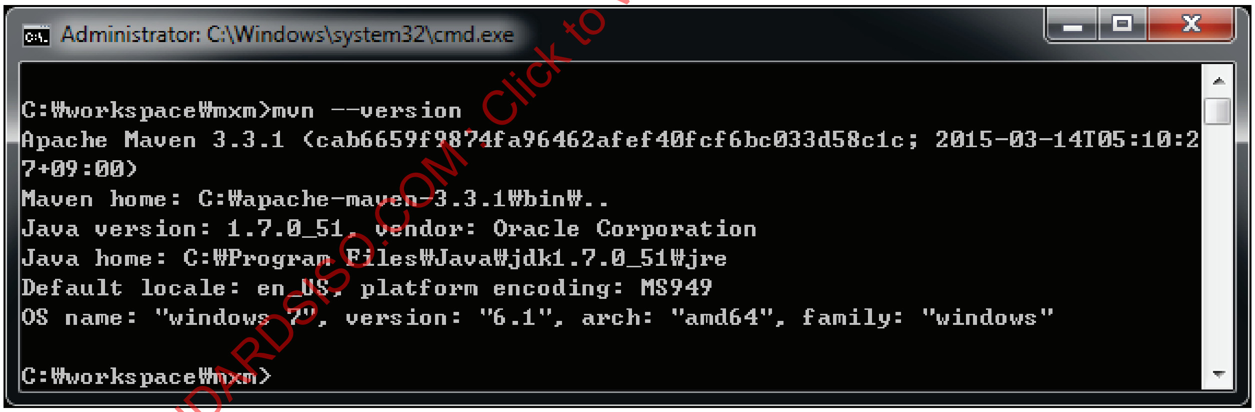
— Graphviz: http://www.graphviz.org/Download.php

B.2 Building the MXM source code in the command line

1) Verify that you have Maven correctly installed by executing (in any folder) the following command:

```
mvn --version
```

If the result what you've got is similar to the following figure, the test was successful.



```
C:₩workspace₩mxm>mvn --version
Apache Maven 3.3.1 (cab6659f9874fa96462afef40fcf6bc033d58c1c; 2015-03-14T05:10:2
7+09:00)
Maven home: C:₩apache-maven-3.3.1₩bin₩..
Java version: 1.7.0_51, vendor: Oracle Corporation
Java home: C:₩Program Files₩Java₩jdk1.7.0_51₩jre
Default locale: en_US, platform encoding: MS949
OS name: "windows 7", version: "6.1", arch: "amd64", family: "windows"

C:₩workspace₩mxm>
```

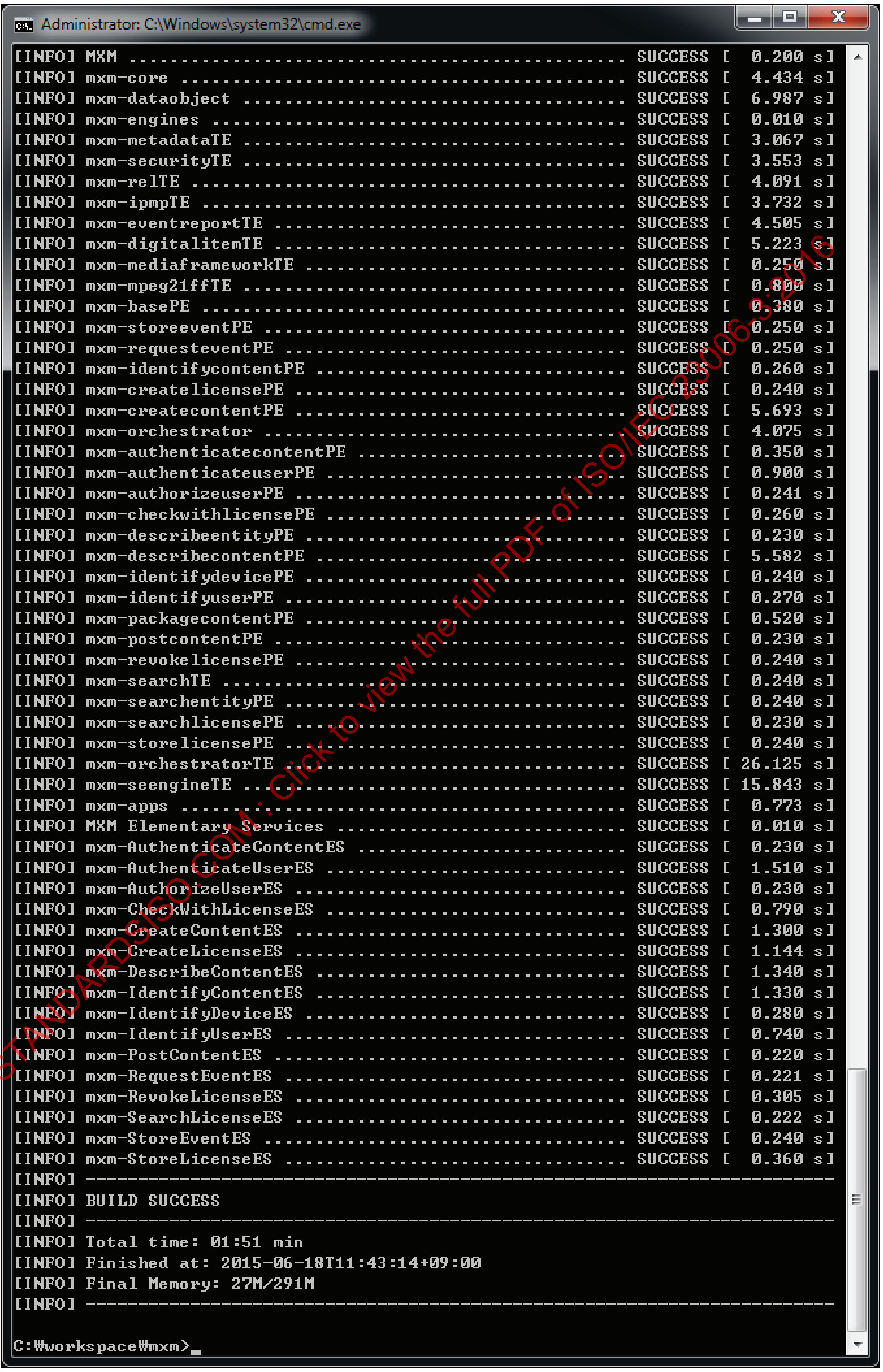
2) Change the directory path of SEM.xml, CIM.xml, and IIM.xml to a proper location. The path parameters are included in the following file:

```
${basedir}\mxm-engines\mxm-seengineTE\src\test\java\iso\mpeg\seengineTE\
test\java\iso\mpeg\seengineTE\test\SEEngineTest.java
```

3) Move to the folder where you have checked out the MXM Java source code and execute the following command:

```
mvn  install
```

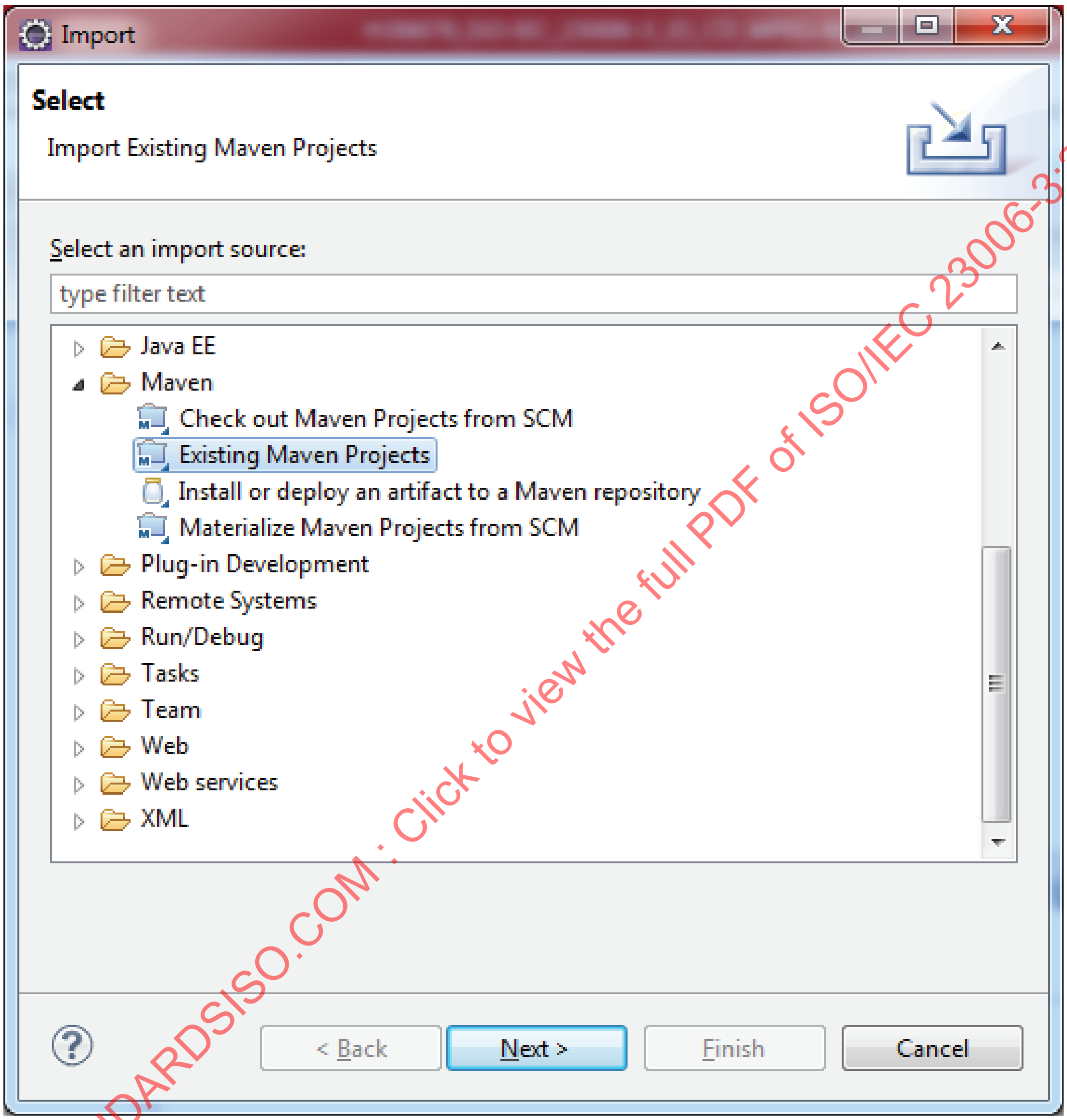
As the result, you should get the same output as shown in the following figure.



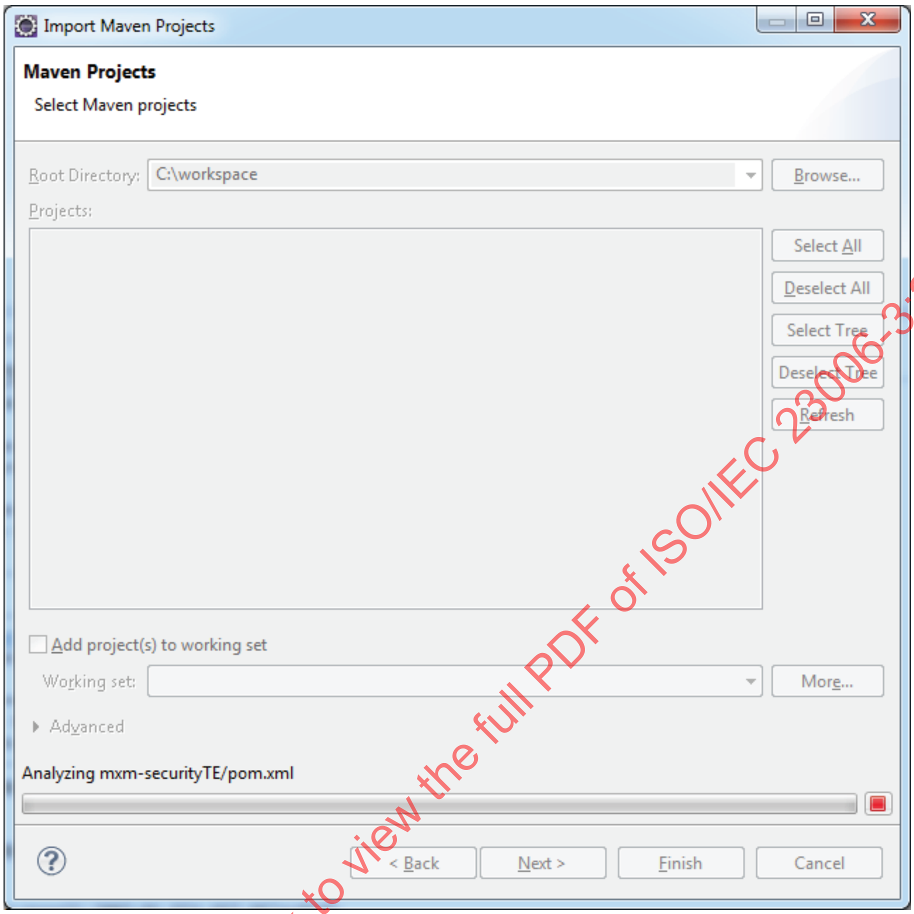
```
[INFO] MXM .............................................. SUCCESS [  0.200 s]
[INFO] mxm-core ......................................... SUCCESS [  4.434 s]
[INFO] mxm-dataobject ................................... SUCCESS [  6.987 s]
[INFO] mxm-engines ...................................... SUCCESS [  0.010 s]
[INFO] mxm-metadataTE ................................... SUCCESS [  3.067 s]
[INFO] mxm-securityTE ................................... SUCCESS [  3.553 s]
[INFO] mxm-relTE ........................................ SUCCESS [  4.091 s]
[INFO] mxm-ipmpTE ....................................... SUCCESS [  3.732 s]
[INFO] mxm-eventreportTE ................................ SUCCESS [  4.505 s]
[INFO] mxm-digitalitemTE ................................ SUCCESS [  5.223 s]
[INFO] mxm-mediaframeworkTE ............................. SUCCESS [  0.250 s]
[INFO] mxm-mpeg21ffTE ................................... SUCCESS [  0.800 s]
[INFO] mxm-basePE ....................................... SUCCESS [  0.380 s]
[INFO] mxm-storeeventPE ................................. SUCCESS [  0.250 s]
[INFO] mxm-requesteventPE ............................... SUCCESS [  0.250 s]
[INFO] mxm-identifycontentPE ............................ SUCCESS [  0.260 s]
[INFO] mxm-createlicensePE .............................. SUCCESS [  0.240 s]
[INFO] mxm-createcontentPE .............................. SUCCESS [  5.693 s]
[INFO] mxm-orchestrator ................................. SUCCESS [  4.075 s]
[INFO] mxm-authenticatecontentPE ........................ SUCCESS [  0.350 s]
[INFO] mxm-authenticateuserPE ........................... SUCCESS [  0.900 s]
[INFO] mxm-authorizeuserPE .............................. SUCCESS [  0.241 s]
[INFO] mxm-checkwithlicensePE ........................... SUCCESS [  0.260 s]
[INFO] mxm-describeentityPE ............................. SUCCESS [  0.230 s]
[INFO] mxm-describecontentPE ............................ SUCCESS [  5.582 s]
[INFO] mxm-identifydevicePE ............................. SUCCESS [  0.240 s]
[INFO] mxm-identifyuserPE ............................... SUCCESS [  0.270 s]
[INFO] mxm-packagecontentPE ............................. SUCCESS [  0.520 s]
[INFO] mxm-postcontentPE ................................ SUCCESS [  0.230 s]
[INFO] mxm-revokelicensePE .............................. SUCCESS [  0.240 s]
[INFO] mxm-searchTE ..................................... SUCCESS [  0.240 s]
[INFO] mxm-searchentityPE ............................... SUCCESS [  0.240 s]
[INFO] mxm-searchlicensePE .............................. SUCCESS [  0.230 s]
[INFO] mxm-storelicensePE ............................... SUCCESS [  0.240 s]
[INFO] mxm-orchestratorTE ............................... SUCCESS [ 26.125 s]
[INFO] mxm-seengineTE ................................... SUCCESS [ 15.843 s]
[INFO] mxm-apps ......................................... SUCCESS [  0.773 s]
[INFO] MXM Elementary Services .......................... SUCCESS [  0.010 s]
[INFO] mxm-AuthenticateContentES ........................ SUCCESS [  0.230 s]
[INFO] mxm-AuthenticateUserES ........................... SUCCESS [  1.510 s]
[INFO] mxm-AuthorizeUserES .............................. SUCCESS [  0.230 s]
[INFO] mxm-CheckWithLicenseES ........................... SUCCESS [  0.790 s]
[INFO] mxm-CreateContentES .............................. SUCCESS [  1.300 s]
[INFO] mxm-CreateLicenseES .............................. SUCCESS [  1.144 s]
[INFO] mxm-DescribeContentES ............................ SUCCESS [  1.340 s]
[INFO] mxm-IdentifyContentES ............................ SUCCESS [  1.330 s]
[INFO] mxm-IdentifyDeviceES ............................. SUCCESS [  0.280 s]
[INFO] mxm-IdentifyUserES ............................... SUCCESS [  0.740 s]
[INFO] mxm-PostContentES ................................ SUCCESS [  0.220 s]
[INFO] mxm-RequestEventES ............................... SUCCESS [  0.221 s]
[INFO] mxm-RevokeLicenseES .............................. SUCCESS [  0.305 s]
[INFO] mxm-SearchLicenseES .............................. SUCCESS [  0.222 s]
[INFO] mxm-StoreEventES ................................. SUCCESS [  0.240 s]
[INFO] mxm-StoreLicenseES ............................... SUCCESS [  0.360 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 01:51 min
[INFO] Finished at: 2015-06-18T11:43:14+09:00
[INFO] Final Memory: 27M/291M
[INFO] ------------------------------------------------------------------------

C:\workspace\mxm>_
```

B.3 Building the MXM source code with eclipse

1) Run eclipse program and select Import in the File menu. Click Maven in the import dialog and then select Existing Maven Projects:



2) Set Root Directory to the directory where the MXM Java source code has been downloaded:



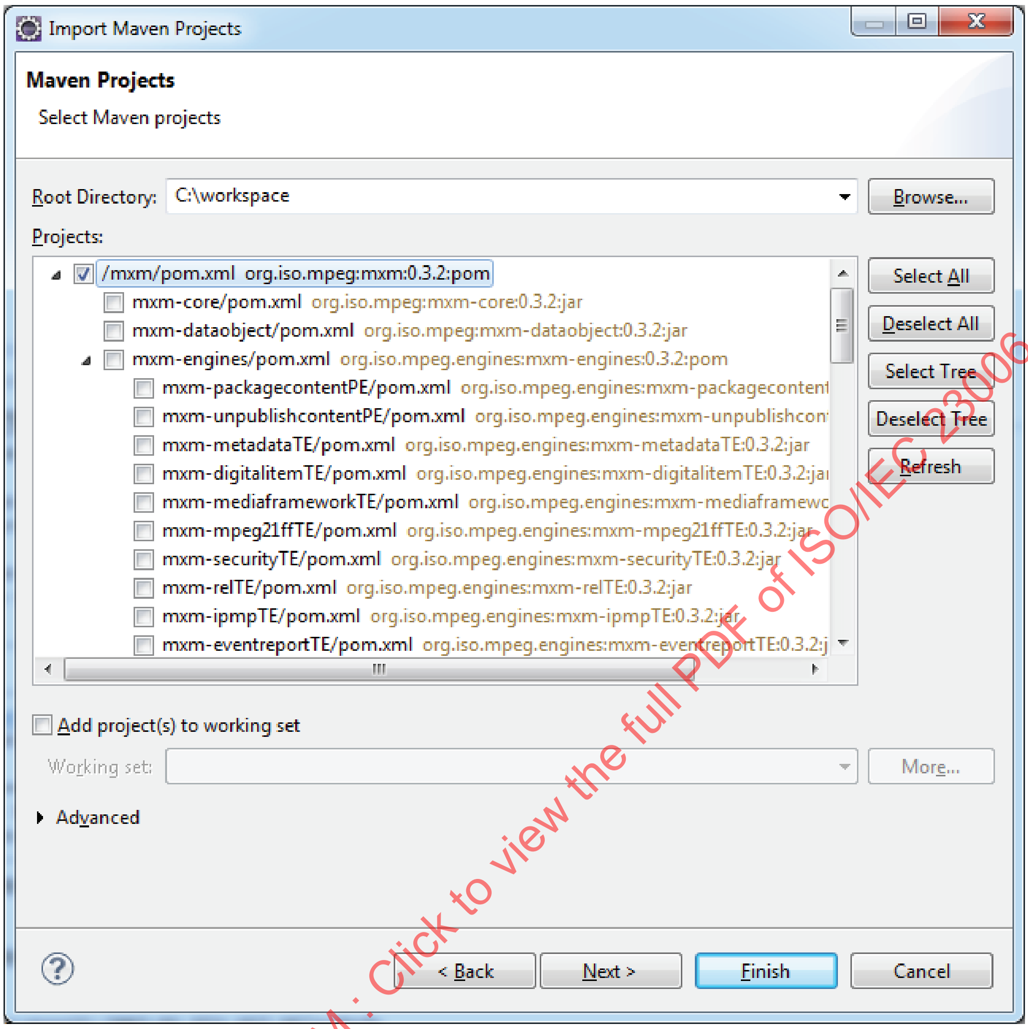
Root Directory: C:\workspace

[ ] Add project(s) to working set

Working set:

Analyzing mxm-securityTE/pom.xml

3) Click Deselect Tree and select /mxm/pom.mxl:



Root Directory: C:\workspace

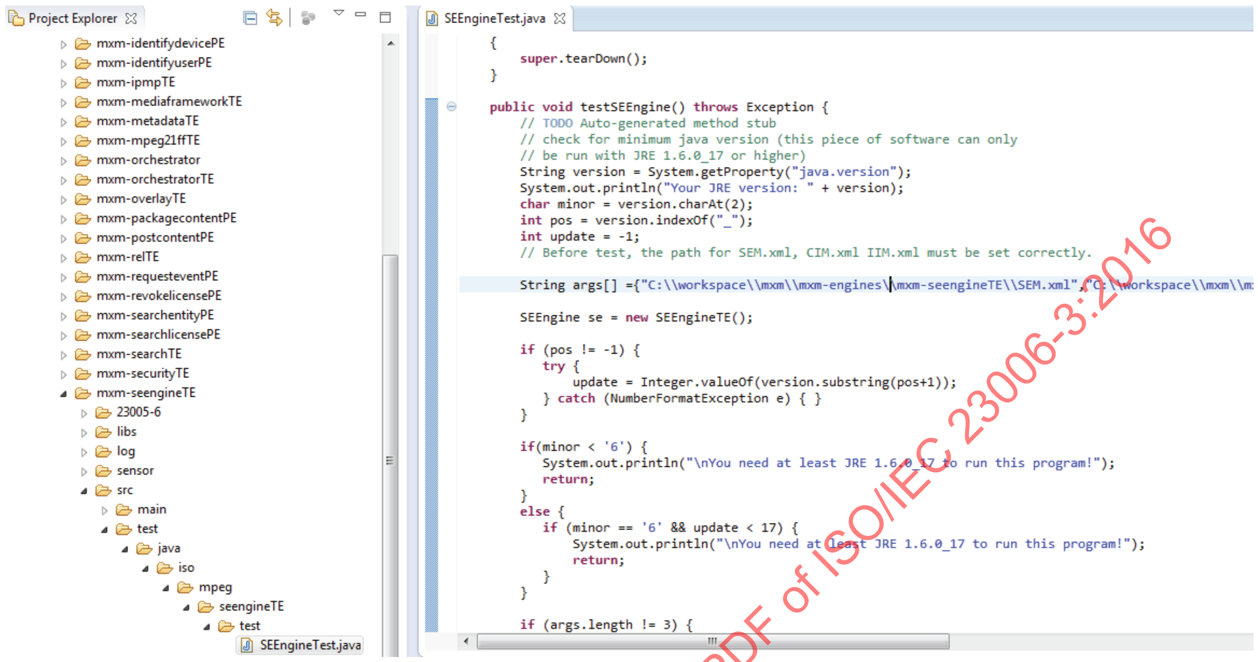
Projects:

```
◢ [✓] /mxm/pom.xml  org.iso.mpeg:mxm:0.3.2:pom
    [ ] mxm-core/pom.xml  org.iso.mpeg:mxm-core:0.3.2:jar
    [ ] mxm-dataobject/pom.xml  org.iso.mpeg:mxm-dataobject:0.3.2:jar
  ◢ [ ] mxm-engines/pom.xml  org.iso.mpeg.engines:mxm-engines:0.3.2:pom
        [ ] mxm-packagecontentPE/pom.xml  org.iso.mpeg.engines:mxm-packagecontent
        [ ] mxm-unpublishcontentPE/pom.xml  org.iso.mpeg.engines:mxm-unpublishcon
        [ ] mxm-metadataTE/pom.xml  org.iso.mpeg.engines:mxm-metadataTE:0.3.2:jar
        [ ] mxm-digitalitemTE/pom.xml  org.iso.mpeg.engines:mxm-digitalitemTE:0.3.2:jar
        [ ] mxm-mediaframeworkTE/pom.xml  org.iso.mpeg.engines:mxm-mediaframewc
        [ ] mxm-mpeg21ffTE/pom.xml  org.iso.mpeg.engines:mxm-mpeg21ffTE:0.3.2:jar
        [ ] mxm-securityTE/pom.xml  org.iso.mpeg.engines:mxm-securityTE:0.3.2:jar
        [ ] mxm-relTE/pom.xml  org.iso.mpeg.engines:mxm-relTE:0.3.2:jar
        [ ] mxm-ipmpTE/pom.xml  org.iso.mpeg.engines:mxm-ipmpTE:0.3.2:jar
        [ ] mxm-eventreportTE/pom.xml  org.iso.mpeg.engines:mxm-eventreportTE:0.3.2:j
```

[ ] Add project(s) to working set

Working set:

4) Change the directory path of SEM.xml, CIM.xml, and IIM.xml:



```
    {
        super.tearDown();
    }

    public void testSEEngine() throws Exception {
        // TODO Auto-generated method stub
        // check for minimum java version (this piece of software can only
        // be run with JRE 1.6.0_17 or higher)
        String version = System.getProperty("java.version");
        System.out.println("Your JRE version: " + version);
        char minor = version.charAt(2);
        int pos = version.indexOf("_");
        int update = -1;
        // Before test, the path for SEM.xml, CIM.xml IIM.xml must be set correctly.

        String args[] ={"C:\\workspace\\mxm\\mxm-engines\\mxm-seengineTE\\SEM.xml","C:\\workspace\\mxm\\m

        SEEngine se = new SEEngineTE();

        if (pos != -1) {
           try {
               update = Integer.valueOf(version.substring(pos+1));
           } catch (NumberFormatException e) { }
        }

        if(minor < '6') {
           System.out.println("\nYou need at least JRE 1.6.0_17 to run this program!");
           return;
        }
        else {
           if (minor == '6' && update < 17) {
               System.out.println("\nYou need at least JRE 1.6.0_17 to run this program!");
               return;
           }
        }

        if (args.length != 3) {
```