INTERNATIONAL STANDARD

ISO/IEC 23006-1

Third edition
2018-05

Information technology — Multimedia service platform technologies —

Part 1: Architecture

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

Partie 1: Architecture

Reference number
ISO/IEC 23006-1:2018(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: 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.

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 document 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 and IEC 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 voluntary nature of standards, 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.

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

This third edition cancels and replaces the second edition (ISO/IEC 23006-1:2013), which has been technically revised.

The main changes compared to the previous edition are as follows:

— A new reference diagram of an MPEG-M device where the middleware is seen as a black box. ISO/IEC 23006-2 specifies a particular instance of the MPEG-M middleware which is organized in engines.

— High level API exposed by any MPEG-M middleware.

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

Introduction

The ISO/IEC 23006 series has been developed to enable the easy design and implementation of media-handling value chains supported by devices that interoperate because they are all based on the same set of technologies, especially MPEG technologies. The functionalities provided by the MPEG technologies are accessible via application programming interfaces (API).

The ISO/IEC 23006 series specifies a service-oriented architecture (Part 1), middleware API (Part 2), conformance and reference software (Part 3), a set of protocols supporting elementary services (Part 4) and the combination of elementary services into aggregated services (Part 5).

MPEG-M supports the service providers' desire to designed and deploy at reduced cost innovative multimedia services. This is achieved by identifying a set of elementary services (ES) and defining the corresponding set of protocols and APIs to enable any user in an MPEG-M value chain to access those services in an interoperable fashion.

NOTE An MPEG-M value chain is a collection of users, including creators, end users and service providers that conform to the ISO/IEC 23006 series.

In many real-world MPEG-M value chains, service providers would not be able to exploit the potential of the series if they were confined to only offer elementary services. Therefore service providers (SP) will typically offer bundles of ESs, known as aggregated services (AS). In general, there will be a plurality of SPs offering the same or partially overlapping aggregated services. For example, a SP offering user description services, may offer content description services as well.

Starting from ISO/IEC 23006-4, an aggregation of services can put together a number of services generating a complex ISO/IEC 23006 value network, having different topologies and associated services.

Using the ISO/IEC 23006 series, a digital media ecosystem can be established, where:

— developers can offer MPEG-M service components to the professional market because a market will be enabled by the standard MPEG-M component service API;

— manufacturers can offer MPEG-M devices to the global consumer market because of the global reach of MPEG-M services;

— service providers can set up and launch new attractive MPEG-M services because innovative MPEG-M value chains can be easily designed and implemented;

— developers can make available a variety of multimedia applications;

— users can seamlessly create, offer, search, access, pay/cash and consume MPEG-M services.

The ISO/IEC 23006 series extends the devices capabilities with advanced features such as content generation, processing, and distribution by a large number of users; easy creation of new services by combining service components of their choice; global, seamless and transparent use of services regardless of geo-location, service provider, network provider, device manufacturer and provider of payment and cashing services; diversity of user experience through easy download and installation of applications produced by a global community of developers since all applications share the same middleware 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, especially MPEG technologies.

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

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

Part 2 — *MPEG extensible middleware (MXM) API*: specifies the middleware APIs;

Part 3 — *Conformance and reference software*: 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 1: Architecture

1 Scope

This document specifies the MPEG-M architecture that is made accessible through the set of MPEG-M high level APIs, MPEG extensible middleware API, elementary services and service aggregation specified in ISO/IEC 23006-2, ISO/IEC 23006-4 and ISO/IEC 23006-5 and as a software implementation in ISO/IEC 23006-3, respectively.

NOTE Annex A provides an informative example of how MPEG-M can be used to create a fully-fledged multimedia platform.

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 23000-16, *Information technology — Multimedia Application Format (MPEG-A) — Part 16: Publish Subscribe Application Format*

ISO/IEC 23006-2, *Information technology — Multimedia service platform technologies — Part 2: MPEG extensible middleware (MXM) APIs*

ISO/IEC 23006-3, *Information technology — Multimedia service platform technologies — Part 3: Conformance and reference software*

ISO/IEC 23006-4, *Information technology — Multimedia service platform technologies — Part 4: Elementary services*

ISO/IEC 23006-5, *Information technology — Multimedia service platform technologies — Part 5: Service aggregation*

3 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
application
software that runs in the environment and makes calls to the high-level API

3.2
computing platform
combination of hardware and basic software, such as operating system and drivers, that executes software

3.3
device
environment that includes software conforming to this specification

3.4
engine
component of the middleware that provides a defined functionality or set of functionalities accessible via API

3.5
environment
combination of hardware and software that exposes high-level API, and interfaces to devices that expose low-level API

3.6
high-level API
abstracted API exposed by a device to enable an application to access its functionalities and services

3.7
low-level API
programmatic interfaces exposed by devices, such as security devices, to the middleware

3.8
middleware
software providing functionalities to local and remote applications

3.9
middleware API
combination of the API of all engines in the middleware

3.10
orchestrator engine
special engine capable of creating chains of engines

EXAMPLE To set-up a sequence of connected engines to execute a high-level API call such as play.

3.11
user
any entity making use of a device

4 Abbreviated terms and elements of the MPEG-M architecture

4.1 Abbreviated terms

API	application programming interface
BPMN	business process model and notation
CEL	contract expression language
DASH	dynamic adaptive streaming over HTTP
DID	digital item declaration

DIDL	digital item declaration language
DII	digital item identification
ER	event report
ERR	event report request
HTTP	hypertext transport protocol
IPMP	intellectual property management and protection
PSAF	publish/subscribe application format
PubSub	publish/subscribe messaging model
REL	rights expression language
URI	uniform resource identifier

4.2 Elements of the MPEG-M architecture

Device	device conforming to this specification
Application	software component that runs on devices
High-level API	abstracted API exposed by a device to enable an application to access its functionalities and services
Middleware	software layer providing functionalities to local and remote applications
Engine	bundle of technologies that provide a defined functionality or set of functionalities
Engine API	API exposed by an engine to enable another engine/application to access its functionality
Mid-level API	collection of all engine API
Low-level API	API exposed by devices such as network, smart card or battery attached to a device
Orchestrator Engine	special engine capable of creating chains of engines to execute a high-level application call such as “play” that typically requires access to multiple engine functionalities

5 Namespace conventions

Throughout this document, 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 document, consistent namespace prefixes are used. Table 1 gives these prefixes and the corresponding namespace.

Table 1 — Namespaces and prefixes

Prefix	Corresponding namespace
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
erl	urn:mpeg:mpeg21:2005:01-ERL-NS

Table 1 *(continued)*

Prefix	Corresponding namespace
fru	urn:mpeg:mpegB:schema:FragmentRequestUnits:2007
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
dii	urn:mpeg:mpeg21:2002:01-DII-NS
mpqf	urn:mpeg:mpqf:schema:2008
mpeg4ipmp	urn:mpeg:mpeg4:IPMPSchema:2002
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
didl-mpegm	urn:mpeg:mpegm:2011:12-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
rel-r	urn:mpeg:mpeg21:2003:01-REL-R-NS
rel-sx	urn:mpeg:mpeg21:2003:01-REL-SX-NS
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance
dsig	http://www.w3.org/2000/09/xmldsig#
xenc	http://www.w3.org/2001/04/xmlenc#

6 System overview

A general architecture of a device is given in Figure 1.

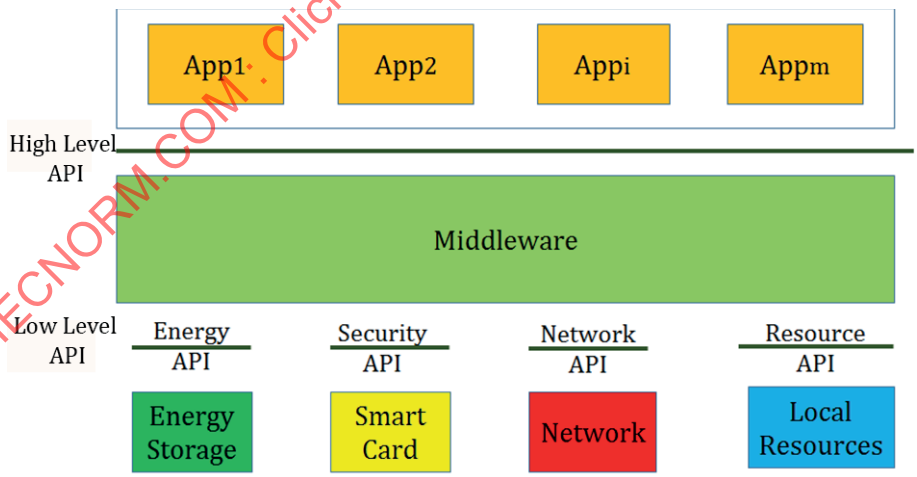

Figure 1 — Generic MPEG-M device architecture

Applications run on the device and perform their expected actions by accessing the middleware functionalities via the high-level API. In general a plurality of applications run on a device (there may be other non-MPEG-M applications but these are not relevant for this document). Some may be "resident", e.g., they have been loaded by the device manufacturer while some may be temporary, e.g., they have been downloaded for a specific purpose.

Examples of applications include:

— Video viewer: an application to view governed videos from a service provider;

— Content creator: an application to create content with audio-visual resources, metadata and rights information;

— Licence server: an application managing a licence-issuing service.

The **high-level API**s are abstracted APIs exposed by a device to enable an application to access its functionalities and services.

The **middleware** allows application easy access to platform functionalities.

Low-level APIs provide access to the functionalities of computing platform and external devices.

ISO/IEC 23006-2 specifies a particular middleware organisation, as depicted in Figure 2.

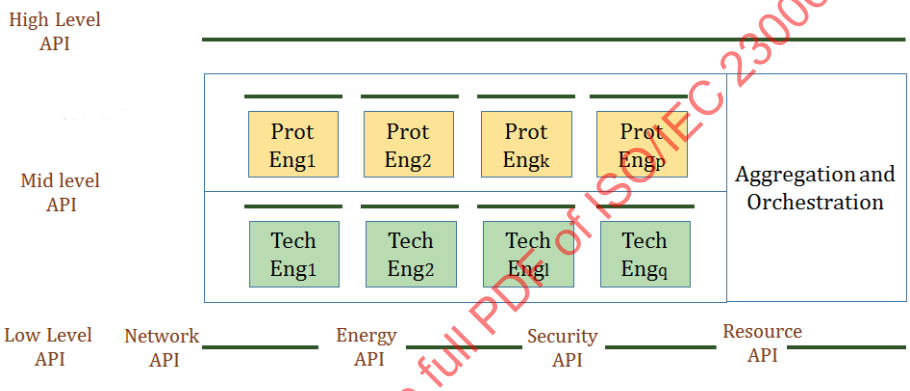

Figure 2 — Engine-based middleware

In this case, the **middleware** is made up of a number of engines (there may be other non-MPEG-M engines but these are not relevant for this specification).

Engines are of two types:

a) Protocol engines (specified in ISO/IEC 23006-4) that implement elementary services. Protocol engines can be combined by aggregation to implement aggregated services.

b) Technology engines (specified in ISO/IEC 23006-2) that implement specific technologies. Technology engines can be combined by orchestration to implement groups of technologies.

Middleware APIs are the combination of the APIs of all engines in the middleware.

Device is a combination of hardware and software conforming to this document. A device is often interfaced to other devices, e.g.:

a) **local resources** that provide computational resources,

b) **security device**, such as a smart card, that performs cryptographic functions,

c) **network** that provide connectivity with other devices, and

d) **energy device** that provides information on battery status.

Two applications running on different networked devices can communicate by executing service protocols, as depicted in Figure 3.

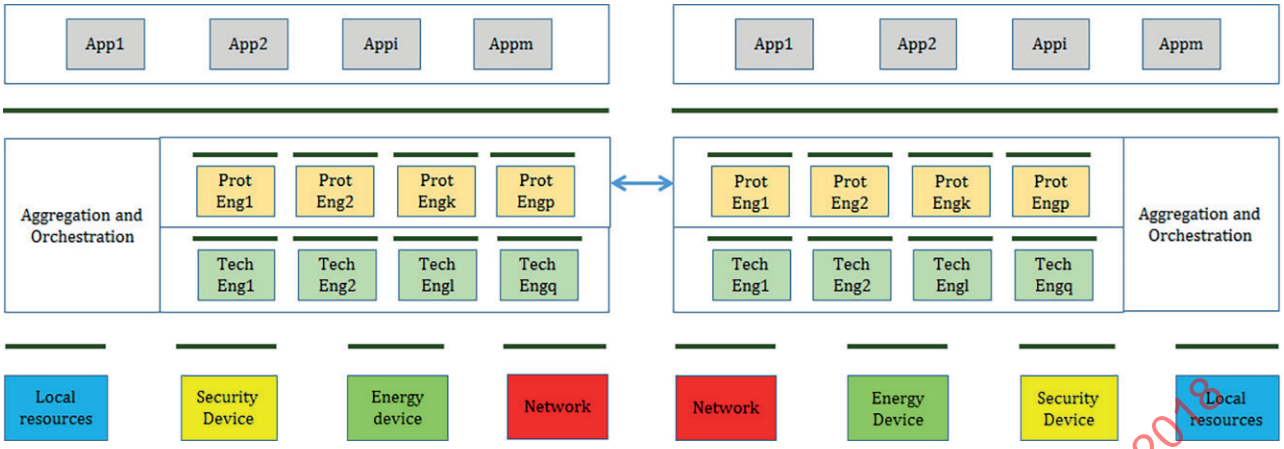

Figure 3 — Communication between two devices

When the device on the right-hand side (e.g., a "client") communicates with the device on the left-hand side (e.g. a "server"), the following happens:

a) A client application makes a service request (e.g., an elementary service such as create licence) using a protocol engine.

b) The corresponding server-side protocol engine, upon receiving the request, calls the appropriate orchestrator engine's API functionality (e.g., REL orchestration) or chain of engines.

c) The orchestrator engine on the server, if required, sets up a chain of engines: in the REL example, just one technology engine (the REL engine) creates the requested licence.

d) The server-side protocol engine returns the licence to the client-side protocol engine.

The same happens if the client application makes an aggregated service request. In this case the orchestrator engine sets up a more complex chain of technology and protocol engines.

When an application is executed, "low-level" calls may be made directly to some engines using the engine API of each specific engine, and high-level API calls like, say, "Play (GovernedContent)" which will be handled by the orchestrator engine. This is depicted in Figure 4.

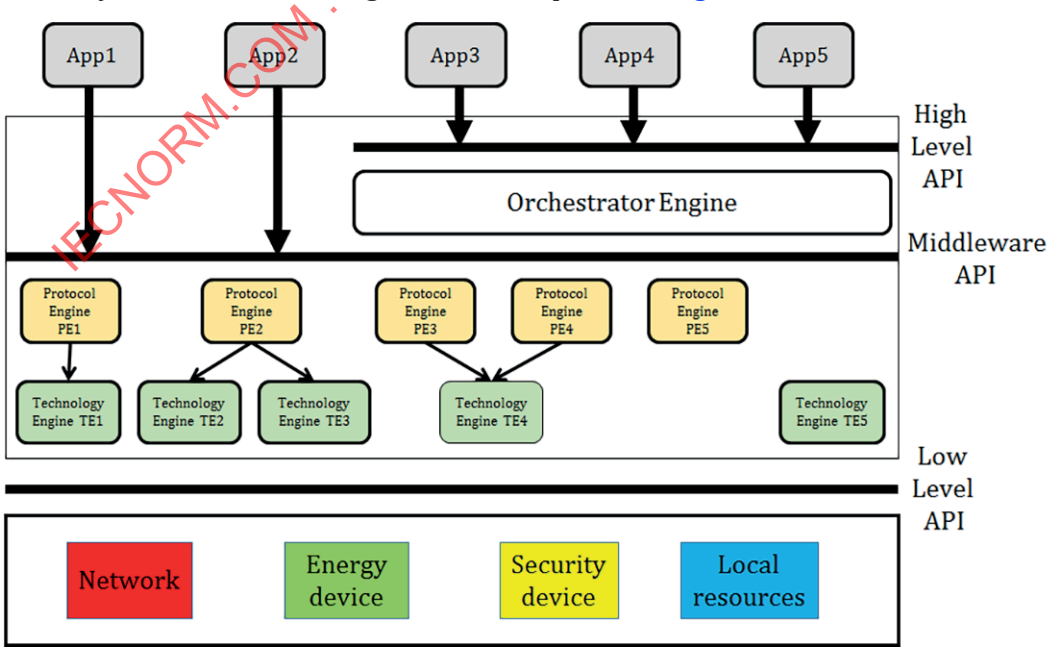

Figure 4 — View inside a device

Making reference to Figure 1, the following possibilities exist for an application:

a) It calls the middleware API to access a protocol engine which in its turn calls a technology engine.

b) It calls the middleware API to access a protocol engine which in its turn calls a plurality of technology engines.

c) It calls the middleware API to access a combination of protocol engines which call a single technology engine.

d) It calls the middleware API to access a protocol engine.

e) It calls the middleware API to access a technology engine.

f) It calls the high-level API to access the orchestrator engine.

The orchestrator engine, by calling the engine APIs of specific engines, is capable of setting up chains of engines for handling complex operations, orchestrating the intervention and send/receive data to/from the particular chain of engines that a given high-level call will trigger, thus relieving applications from the need to carry the logic of handling them. Each engine will contain a specific set of technologies accessible by an application, the orchestrator engine and any other engine, by means of its own engine API.

For instance, in the case of "Play (GovernedContent)" the orchestrator engine could set-up the following chain:

a) MP21 file engine (e.g. open the file and extract the digital item);

b) Digital item engine (e.g. extract metadata and rights information);

c) REL engine (e.g. verify if the right to play is granted);

d) IPMP engine (e.g. set up IPMP tools to decrypt protected resources);

e) Security engine (e.g. initialise the IPMP tools with decryption keys);

f) Metadata engine (e.g. for presentation of content metadata to the user);

g) Media framework engine (e.g. demux, decode and render audio-visual resources);

h) and possibly others.

It should be noted that only the APIs of an engine are normative; how each engine handles the operations needed to carry out a request depends on the specific software or hardware engine implementation.

Figure 5 depicts the general case of an aggregated service implemented by the aggregation of three protocol engines (PE_a, PE_b and PE_c), the first calling a single technology engine, the second three TEs and the third two TEs.

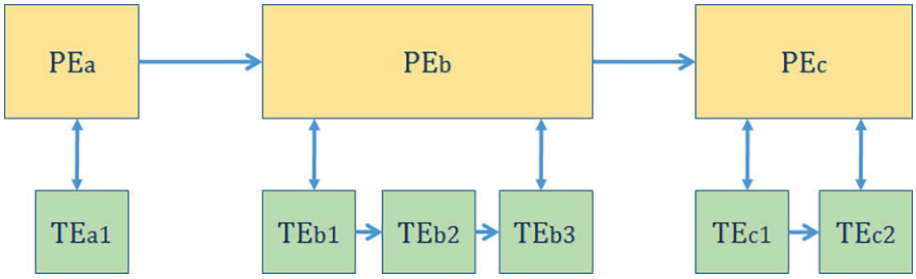

Figure 5 — Engine aggregation and orchestration

7 MPEG-M architecture

7.1 General

MPEG-M specifies an architecture containing MPEG-standard multimedia technologies whose purpose is to enable the easy design and implementation of media-handling value chains whose devices interoperate because they are all based on the same set of technologies exposed through standard APIs.

The elements of the architecture are:

a) **Application**, software that makes calls to the high-level API;

b) **High-level API**, exposed to applications;

c) **Middleware**;

d) **Low-level API**, exposed by computing platform and external devices;

e) **Device**, a combination of hardware and software conforming to ISO/IEC 23006.

In case of engine-based middleware the following elements should be added:

a) **Engine API** that can be used to access engine functionality;

b) **Engines**, collections of specific technologies that are bundled together to provide a specific functionality that is provided to applications;

c) **Orchestrator engine**, a special engine capable of creating chains of engines to execute a high-level application call such as “Play”;

d) **Middleware API**, the collection of all engine APIs.

The engine APIs are divided in three categories.

— **Creation API**: this includes APIs to create data structures, files, elementary streams, etc. conforming to the respective standards.

— **Access API**: this includes APIs to parse data structures, files, decode elementary streams, etc. in order to retrieve the information contained within.

— **Engine-specific APIs**: this includes specific APIs of an engine.

The first two categories include those recurring for more than one engine, while the last category includes those APIs which are specific for one engine.

7.2 High-level API

7.2.1 General

The high-level APIs are abstracted APIs exposed by a device to enable an application to access its functionalities and services high-level API for publish/subscribe pattern.

The publish/subscribe messaging service can be realized with a topic-based, content-based or mixed filtering. A publish-subscribe messaging service architecture shall implement the following methods:device.

Methods

a) Method: Publish.

Publish content with metadata and/or on a particular topic.

— param: Publication that is the publication type of payload as specified in ISO/IEC 23000-16.

— return: Publication status.

b) Method: Subscribe.

Subscribe an interest for a particular content and/or on a particular topic.

— param: Subscription that is the subscription type of payload as specified in ISO/IEC 23000-16.

— return: Subscription status.

c) Method: Store.

Store content.

— param: Resource that is the resource type of payload as specified in ISO/IEC 23000-16.

— return: digital item identifier of resource.

d) Method: Play.

Play content.

— param: Resource that is the resource type of payload as specified in ISO/IEC 23000-16

— return: Nothing

7.2.2 Network services

These APIs are used to configure network service conditions, such as selection of network service operator, service quality etc. Table 2 shows a list of Network services API.

Table 2 — Network service API

Service operation	Input	Response	Description
GetNetworkTypeList()		Available network types	Get a list of available network types.
SetNetworkType()	Network type ID	OK/Fail	Choose network type from a list of available network types.
GetServiceProviderList()		Available service provider IDs	Get a list of available service providers. Service provider codes are externally defined.
SetServiceProvider()	Service provider ID	OK/Fail	Choose network service operator from a list of available service providers.

7.2.3 Energy management

This API enables to obtain energy related information and use of energy saving functionality. Application developers do not have to consider detailed energy saving strategy, but just consider when, and in which situation, energy saving functionality should be activated. Table 3 shows the set of energy management API.

Table 3 — Energy management API

Service operation	Input	Response	Description
SetEnergyStrategy()	List of state transition condition (e.g., battery level 40%, 20%, ...) List of operating conditions (e.g., Level 1, Level 2, ...) [Optional: function priority (e.g., video quality, video frame rate...)]	OK/Fail	Set a power consumption strategy (e.g., the display power consumption type) to the middleware. The middleware monitors battery charge level and sends the device level power commands to the local resource/device such as CPU, display in order to save the power consumption when the battery charge level is getting lower than a specified state transition condition.
GetBatteryChargeLevel()	[Optional: accuracy, device identifier]	BatteryChargeLevel	Get: a) current battery charge level, and its accuracy if requested; b) battery charge level of connected devices identified by device identifier.

7.2.4 Security

The methods exposed are described in Table 4.

Table 4 — Security API

Service operation	Input	Response	Description
VerifyIdentity()	Credentials	OK/Fail	Verify the credentials in order to permit access to device functionalities.
VerifyResource()	Resource	OK/Fail	Verify the authenticity and integrity of, and decrypt, a resource.
SecuritizeResource()	Resource	OK/Fail	Sign and encrypt a resource.

7.3 MPEG-M middleware

7.3.1 Protocol engines

Protocol engines are implementations of elementary services. A list is provided in Table 5.

Table 5 — Elementary services

Elementary service	Allows users to
Authenticate content	Confirm the identity of a content item
Create content	Create content remotely
Deliver content	Transfer content between users of an AIT value chain
Describe content	Associate metadata to content
Identify content	Assign identifiers to content
Package content	Make content ready for delivery
Post content	Let other users access their content
Process content	Perform operations on content

Table 5 *(continued)*

Elementary service	Allows users to
Request content	Retrieve content
Revoke content	Revoke availability of content
Search content	Search for content
Store content	Save content for later use
Transact content	Transact content
Authenticate contract	Confirm the identity and signers of a contract
Check with contract	Verify if a usage request matches with the content (e.g., obligations, prohibitions) expressed in a contract
Create contract	Generate a contract
Deliver contract	Transfer contract between users of an AIT value chain
Identify contract	Assign identifiers to contract
Negotiate contract	Achieve an agreement on the terms and conditions of use, e.g., obligations and prohibitions, with respect to a content item, a device or a service with other users
Present contract	Understand a contract
Request contract	Request a contract
Revoke contract	Discontinue the validity of a contract
Search contract	Search for a contract
Store contract	Save contract for later use
Verify contract	Check the integrity of a contract
Describe device	Associate metadata to devices of an AIT value chain
Identify device	Assign a unique identifier to devices of an AIT value chain
Request device	Request a device
Search device	Search for a device
Verify device	Check the integrity of a device
Request event	Request the creation of an event report for one or more events that have occurred or are to occur
Store event	Store an event
Authenticate licence	Confirm the identity and issuance of a licence
Check with licence	Obtain authorization of a usage request according to rights expressed in a licence
Create licence	Generate a licence
Identify licence	Assign identifiers to licence
Negotiate licence	Achieve an agreement on the terms and conditions of use, e.g., rights and conditions of a content item, or a service
Present licence	Understand licence
Process licence	Change the content of a licence
Request licence	Request a licence
Revoke licence	Discontinue the validity of a licence
Search licence	Search for a licence
Store licence	Save licence for later use
Transact licence	Transact a licence
Verify licence	Check the integrity of a licence
Describe service	Associate metadata to services of an AIT value chain
Search service	Search for a particular service
Authenticate user	Authenticate users
Authorise user	Obtain authorization of some usage(s) by the user

Table 5 *(continued)*

Elementary service	Allows users to
Describe user	Describe users of an AIT value chain
Identify user	Assign unique identifiers to users
Search user	Search for another user

7.3.2 Technology engines

The list of technology engines is provided in Table 6.

Table 6 — Technology engines and classification of their APIs

No.	Engine	Creation API	Access API	Engine-specific API
1.	CDVS	Y	Y	
2.	CEL	Y	Y	— Search — Check with
3.	Digital item	Y	Y	
4.	Event reporting	Y	Y	— Register events — Transmit event reports
5.	Green metadata	Y	Y	
6.	IPMP	Y	Y	— IPMP tool instantiation — IPMP tool initialisation
7.	Media framework	Y	Y	
8.	Metadata	Y	Y	
9.	MPEG-21 file format	Y	Y	
10.	Overlay	Y	Y	— Store/retrieve messages — Propagate message
11.	REL	Y	Y	— Validation — Authorization
12.	Search	Y	Y	
13.	Security	Y	Y	— Authentication — Integrity
14.	Sensory effects	Y	Y	— Manipulate a sensory effect metadata (SEM) structure; — Retrieve by identifier or object; — Adapt to get device capabilities and user preferences; — Actuate to set device commands of the actuator

NOTE Where no engine-specific API is defined the corresponding MPEG-M engine has no engine-specific API.

7.4 Orchestration

The case analysed in this subclause shows the operation of the orchestrator technology engine in a particularly articulated case. If PubSub is used to access resources, when a subscriber has completed a PubSub workflow (i.e., has received the notification that a match of his subscription with a publication has been found), the user clicks on the notification. The workflow unfolds based on Figure 6.

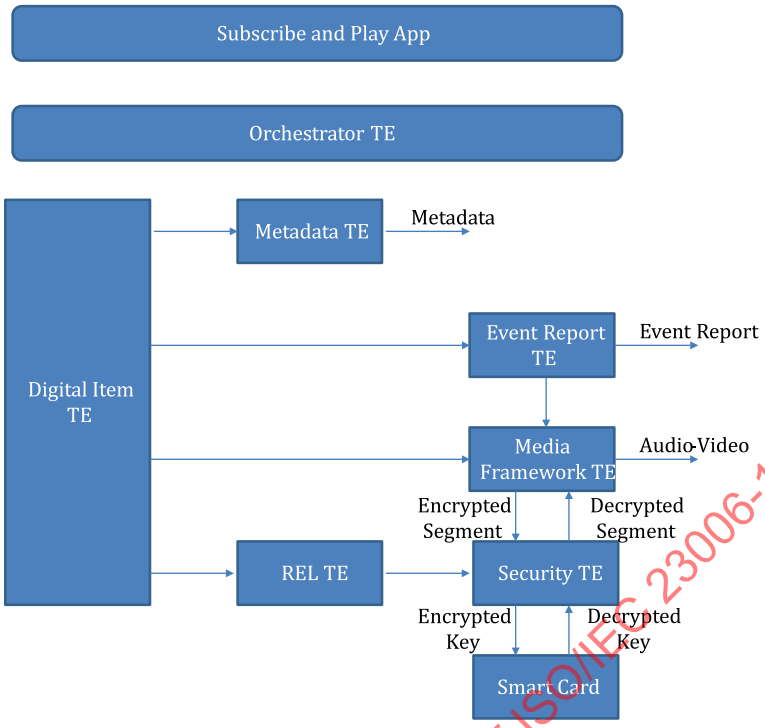

Figure 6 — Operation of orchestrator in resource playing

a) Subscriber user

   1) Parses notification

   2) Requests network technology engine to get resource information

   3) Receives resource information

   4) Decides to see what the video is about

   5) Presses preview button on subscribe and play app

b) Subscribe and play app requests video metadata

c) Digital item technology engine

   1) Parses digital item

   2) Passes metadata to metadata technology engine

d) Metadata technology engine

   1) Parses metadata

   2) Passes metadata to subscribe and play app

e) If user decides to view video subscribe and play app requests MPEG extensible middleware to play video

f) Digital item technology engine

   1) Continues parsing digital item

2) Passes REL licence to REL technology engine

g) REL technology engine

1) Parses REL licence

2) Passes encrypted decryption key to security technology engine

h) Security technology engine passes encrypted decryption key to smart card

i) Smart card

1) Tries to decrypt encrypted decryption key

2) If successful

i) Passes key to security engine

ii) Informs orchestrator technology engine

3) Otherwise informs orchestrator technology engine of failure

j) If decryption key is successfully decrypted, digital item technology engine

1) Parses digital item

2) Sends resource identification to media framework technology engine

3) Parses event report request

4) Sends event report request to event report technology engine

k) Media framework technology engine

1) Requests network technology engine to get MPD

2) Receives MPD

3) Parses MPD

4) Requests network technology engine to get first segment

5) Receives first segment (encrypted)

6) Passes first segment to security technology engine

l) Security technology engine

1) Decrypts first segment using decryption key (in clear)

2) Passes decrypted first segment to media framework technology engine

m) Event report technology engine

1) Parses event report request

2) Dispatches event report to listed users

n) Media framework technology engine

1) Decodes first segment

2) Writes decoded first segment to decoder buffer

The process continues. However, event reports are sent only after the first segment has been successfully decrypted (of course other criteria are possible, e.g., only when the entire video has been played).

7.5 Aggregated services

Aggregated services (specified in ISO/IEC 23006-5) express a process flow realizing a specific task as well as a new elementary service. A service provider can expose elementary service or aggregated services, constructed from several other elementary services and aggregated services. Since service aggregation is a key point of MPEG-M, BPMN (specified in OMG BPMN 2.0[4]) has been adopted because it allows efficient description of service interactions. Moreover, many different aggregation topologies (not just only a serial version of aggregation) and interaction contacts among services can be quite easily illustrated employing the BPMN graphical notation.

7.6 Reference software and conformance

Reference software and conformance (specified in ISO/IEC 23006-3) describes the reference software and conformance profiles implementing the normative clauses of ISO/IEC 23006-1, ISO/IEC 23006-2, ISO/IEC 23006-4 and ISO/IEC 23006-5.

Annex A
(informative)

MPEG-M based advanced multimedia platform

A.1 General

This annex provides an overview of requirements, functionalities and corresponding MPEG-M based technical solutions for an advanced semantic and content-centric multimedia platform[1)], which shows the ability of the MPEG-M standard to support highly innovative and emerging trends in content distribution.

The platform supports its users in the creation, retrieval, manipulation and consumption of multimedia content represented by digital items, namely:

— Description of resources with metadata extracted from well-known or custom taxonomies of concepts;

— Publication of information on resources into an overlay of peers that arranges its topology based on the very same semantic models;

— Search for resources into focused regions of this semantic overlay, also in cases when a degree of heterogeneity exists between requested resources and their descriptions;

— Fetching and delivering matching content to peers using a content-centric transport of resources at the network level of the platform.

The following terms and definitions are used throughout this annex.

Content-centric networking network that replaces the concept of host address with that of "name of a content"

Peer device capable of acting both as service provider's device and as end-user device

Peer-to-peer (P2P) distributed architecture that partitions tasks or workloads between peers

Publication request to the platform to inform an identified subset of users that a digital item is available

Scalability ability of the platform to accommodate a growing number of users and content with a linear impact on performance

Semantic metadata type of metadata based on standard as well as custom, user-created, ontologies and taxonomies

Subscription request to the platform to issue asynchronous notifications of all existing and future publications that match a user's criterion

1) See http://www.ict-convergence.eu/

A.2 Requirements

Security and trust

— The platform shall ensure unique and universal identification and authenticity of machines/peers, services, digital content and users.

— The platform shall ensure privacy and non-tampering of exchanged data.

— The security core of the platform shall not be able to forge the origin identity of users' actions, digital content and communications.

— The security core of the platform shall be flexible enough to configure user-perceived security and privacy between the extremes of fully authenticated operations (so that trusted business models can be implemented where users know what they are buying, vendors know who users are), and a high degree of privacy and anonymity in the services deployed.

Semantic support

— The platform shall ensure the capability, for its users, of describing multimedia content with semantic metadata, i.e. based on ontologies and taxonomies of concepts.

— The platform shall allow users to employ such ontologies to link content with other content, establishing semantic chains of similarity, affinity, extension etc., between elements of multimedia content.

— The platform shall embed additional knowledge about such ontologies and taxonomies, in order to assert equivalence of concepts between similar conceptual domains, by means of dictionaries that help translating from one semantic model to another.

— The platform shall enable searching for content based on descriptions and usage of dictionaries to expand search queries and obtain high quality matches.

Content-centric operation support

— The platform shall enable distribution of content and signalling messages based on a content-centric networking technology.

— The platform shall ensure a high level of scalability by a decentralized and P2P management of the network between devices in the platform.

Publish/subscribe operation support

— The platform shall enable users to publish content and subscribe to content.

Digital forgetting

— Users shall be able to remove content from the platform without leaving traces of its prior existence.

— User shall be able to delete previous subscriptions without leaving traces of them.

A.3 MPEG-M based solution

The platform is composed of a set of interconnected peers. A complete architecture of a peer, built upon the core MPEG-M architecture, conforms to Figure 3. Each of the 3 layers has its own structure and communicates with other layers via standard APIs.

Applications provide users with the means to create, process and consume multimedia content and digital resources. Special re-usable app elements called tools are defined, so as to facilitate re-use of code in apps.

Middleware is the layer responsible for creating, retrieving, manipulating and consuming digital items and their components. Digital items are published in the middleware, so that they can be found via semantic search operations and delivered to users requesting, searching or subscribing to them.

Computing platform hosts specialized network and security modules, as well as interfaces to the local resources such as file-system and processing power.

Digital item (DI)

The content-centric paradigm of the multimedia platform revolves around the MPEG-21 digital item technology. A DI may contain:

a) Unique and persistent identifiers;

b) Semantic links to other DIs;

c) Resources;

d) Semantically-rich metadata describing resources;

e) Licences expressing rights pertaining the manipulation of resources;

f) Event reporting requests (ERR) which instruct peers to issue event reports (ER) to specific target users/peers if specific actions (e.g. play/store) are performed;

g) Others.

The platform supports three main types of DI enabling key functionalities, such as publish/subscribe:

— Resource (R-DI),

— Publication (P-DI), and

— Subscription (S-DI)

Semantic overlay and dictionaries

The platform allows for a semantic organisation of peers in a virtual overlay network of “fractals”, which are dynamically shaped and connected, on the basis of users’ interests.

Peers join or leave a fractal based on what their users currently publish or subscribe, because users categorize published or subscribed content by referring to ontologies that describe a domain of interest.

A core ontology is devised that provides the basic structuring of the overlay into a hierarchical set of fractals, as shown in Figure A.1.

This semantic overlay is managed by overlay technology engine and post-content protocol engines.

Another engine for key semantic functionalities of the platform is the community dictionary service (CDS) technology engine. It maintains *dictionaries* that help translate concepts and properties from one ontology model to another.

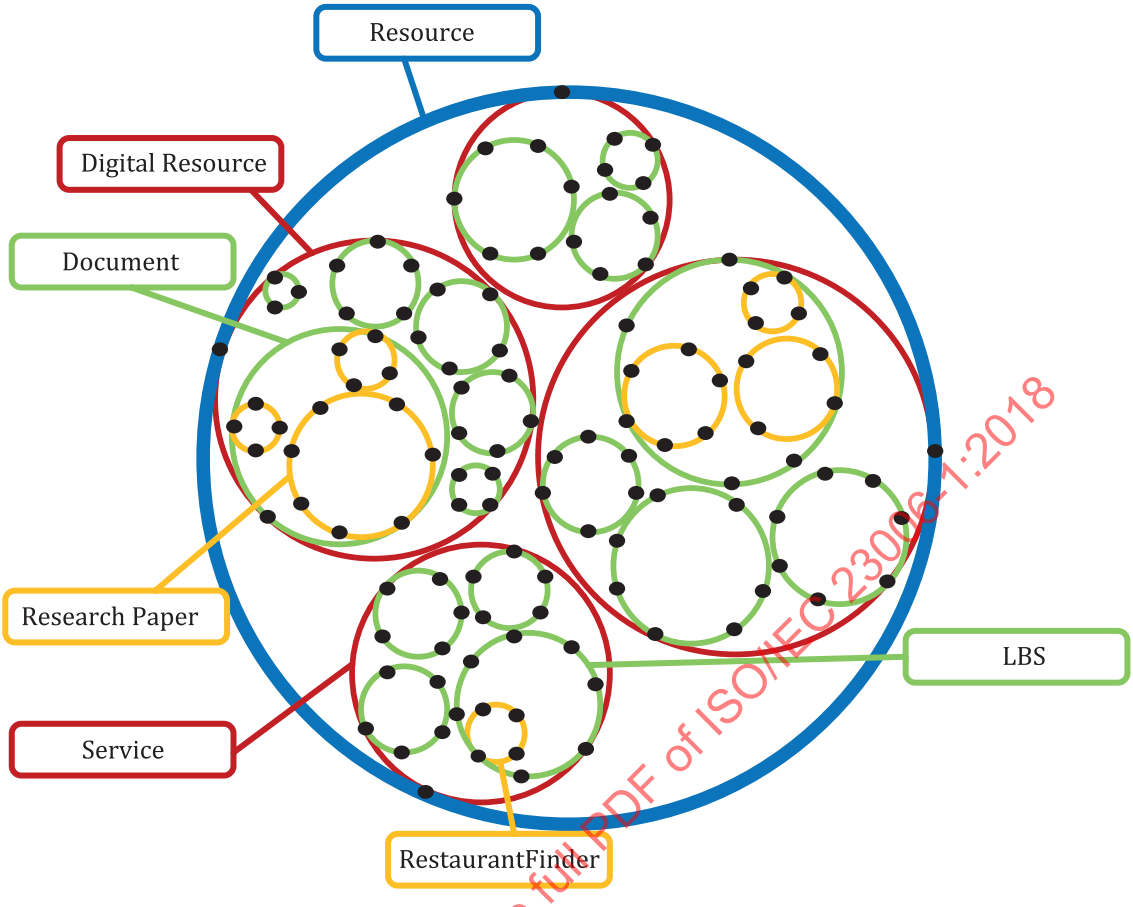

Figure A.1 — Example structuring of the overlay into fractals from the core ontology

The CDS comes into play when users describe resources via the describe content protocol, and when the system performs a match between what is being published and what is being requested or subscribed to, when the knowledge from dictionaries is necessary to translate between concepts and enable the publication-to-subscription match-making.

It is to be noted that the overlay technology engine and the CDS technology engine are platform-specific engines developed in accordance with the MPEG-M standard.

Publish subscribe operations

The subscribe operation in the platform is carried out by invoking a subscribe content service. The publish operation, similarly, is carried out by a publish content service. Publish and subscribe are aggregated services because both publication and subscription are complex operations, which involve a chain of elementary services to be set up and run.

The subscription process can be split into three parts:

— inserting a semantic subscription system wide;

— matching a subscription once relevant content is published;

— delivering a notification to the entity specified by the subscriber.

This process in implemented based on the event reporting standard, in conjunction with the concept of a subscription DI. The MPEG-21 event notification scheme is employed in a distributed and system-wide manner as the event reporting requests are disseminated, throughout the system, by means of their containment within the subscription DIs. A new verb representing the matching of a publication with an outstanding subscription is used.