INTERNATIONAL STANDARD

ISO/IEC 23004-1

First edition
2007-09-15

Information technology — Multimedia Middleware —

Part 1: Architecture

Technologies de l'information — Intergiciel multimédia —

Partie 1: Architecture

Reference number
ISO/IEC 23004-1:2007(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 23004-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 23004 consists of the following parts, under the general title *Information technology — Multimedia Middleware*:

— *Part 1: Architecture*

— *Part 2: Multimedia application programming interface*

— *Part 3: Component model*

— *Part 4: Resource and quality management*

— *Part 5: Component download*

— *Part 6: Fault management*

— *Part 7: System integrity management*

Introduction

MPEG, ISO/IEC JTC 1/SC 29/WG 11, has produced many important standards (MPEG-1, MPEG-2, MPEG-4, MPEG-7, and MPEG-21). MPEG feels that it is important to standardize an application programming interface (API) for Multimedia Middleware (M3W) that complies with the requirements found in the annex to the Multimedia Middleware (M3W) Requirements Document Version 2.0 (ISO/IEC JTC1/SC 29/WG 11 N 6981).

The objectives of MPEG Multimedia Middleware (M3W) are to allow application software to execute multimedia functions with a minimum knowledge of the inner workings of the multimedia middleware, and to allow the triggering of updates to the multimedia middleware to extend the API. The first goal can be achieved by standardizing the API that the multimedia middleware offers. The second goal is much more challenging, as it requires mechanisms to manage the multimedia middleware components, and to ensure that these updates can be integrated in a controlled and dependable manner.

This part of ISO/IEC 23004 provides the following:

- a *vision* for a multimedia middleware API framework that enables
  - application software to control and extend multimedia middleware in a standardized manner;
  - multimedia software to be easily developed for, and deployed across, a variety of platforms;
  - the transparent and augmented use of multimedia resources across a wide range of networks and devices, to optimize the perceived quality for users;
- a method to facilitate the integration of APIs to software components and services in order to harmonize *technologies* for the creation, management, manipulation, transport, distribution and consumption of content;
- a *strategy* for achieving a multimedia API framework by the development of specifications and standards based on well-defined functional requirements through collaboration with other bodies.

Information technology — Multimedia Middleware —

Part 1: Architecture

1 Scope

This part of ISO/IEC 23004 defines the architecture of the MPEG Multimedia Middleware (M3W) technology.

2 Organization of this document

The remainder of this part of ISO/IEC 23004 is structured as follows. Clause 3 gives an overview of the references that are indispensable for the application of this part of ISO/IEC 23004. Clause 4 gives an overview of the terms and definitions used in this part of ISO/IEC 23004.

Clause 5 describes the high level architecture of a complete M3W system. The M3W middleware is part of an M3W system, and ISO/IEC 23004-2 specifies the application programming interface (API) of M3W as well as the realization technology. Subclause 5.2 contains a description of the context of M3W (an M3W system). This subclause also introduces the distinction between M3W API specification and realization of M3W. Subclause 5.3 gives an overview of the M3W API specification. Subclause 5.4 gives an overview of the M3W realization technology that is specified in ISO/IEC 23004-3, ISO/IEC 23004-4, ISO/IEC 23004-5, ISO/IEC 23004-6 and ISO/IEC 23004-7. Subclause 5.5 briefly discusses realization of the M3W, and emphasizes that developers can differentiate their software by producing different realizations.

This part of ISO/IEC 23004 has the following annexes.

Annex A, API specifications reader's guide, explains how the functional and the support parts of the API are specified.

Annex B, Basic types and constants, gives an overview of the basic types and constants that are used in the API specification and the realization technology.

Annex C, API evolution rules, lists the rules for the evolution of API specifications.

Annex D, Naming conventions, lists the naming conventions used in the API specifications and the realization technologies.

Annex E, Constraints on execution architecture. In the API specification a number of assumptions are made on the execution architecture. This annex lists the assumptions which hold, unless specified otherwise.

Annex F, Error handling, describes the default error handling mechanism in M3W.

Annex G, Notification, describes the default notification mechanism in M3W.

Annex H, Get set patterns, describes the default way of dealing with 'get set' patterns in M3W.

Annex I, Handling variation, explains how to deal with variation in M3W systems.

Annex J, API qualifiers, contains a table that lists all of the qualifiers that are used in the API specification.

Annex K, Name abbreviations guide, gives an alphabetical list of words and their abbreviations commonly used in names.

Annex L, IDL, describes the two variations of IDL used in ISO/IEC 23004. One variation is used for the specification of the M3W API, the other one is used in the realization technology. This annex explains how the IDL used to specify the M3W API can be translated into the IDL used in the realization technology.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 23004-2, *Information technology — Multimedia Middleware — Part 2: Multimedia API*

ISO/IEC 23004-3, *Information technology — Multimedia Middleware — Part 3: Component model*

ISO/IEC 23004-4, *Information technology — Multimedia Middleware — Part 4: Resource and quality management*

ISO/IEC 23004-5*, Information technology — Multimedia Middleware — Part 5: Component download* 1)

ISO/IEC 23004-6*, Information technology — Multimedia Middleware — Part 6: Fault management* 1)

ISO/IEC 23004-7*, Information technology — Multimedia Middleware — Part 7: System integrity management* 1)

4 Terms and definitions

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

4.1 Specification terms and definitions

4.1.1
API specification
M3W API specification, which defines a collection of software interfaces providing access to coherent streaming-related functionality

4.1.2
interface suite
collection of mutually related interfaces providing access to coherent functionality

4.1.3
logical component
coherent unit of functionality that interacts with its environment through explicit interfaces only

4.1.4
role
abstract class, i.e. a class without implementation defining behavior only

4.1.5
role instance
object playing a role, i.e. an object displaying the behavior defined by the role

1) To be published.

4.1.6
attribute
instance variable associated with a role

NOTE Attributes are used to associate state information with roles.

4.1.7
signature
definition of the syntactic structure of a specification item such as a type, interface or function in IDL

NOTE For C functions, signature is equivalent to prototype.

4.1.8
specification item
entity defined in a specification

NOTE Data type, role, attribute, interface and function are examples of specification items.

4.1.9
IDL
Interface Definition Language

4.1.10
qualifier
predefined keyword representing a property or constraint imposed on a specification item

4.1.11
constraint
restriction that applies to a specification item

4.1.12
execution constraint
constraint on multi-threaded behavior

4.1.13
model type
data type used for specification (modeling) purposes only

NOTE Set, map and entity are examples of model types.

4.1.14
model constant
constant used for specification (modeling) purposes only

4.1.15
enum element type
enumerated type whose values can be used to construct sets (bit vectors) of at most 32 values by logical or-ing

4.1.16
enum set type
32-bit integer data type representing sets of enumerated values

4.1.17
set type
data type whose values are mathematical sets of values of a specific type

NOTE Unlike enum sets, these sets may be infinite.

4.1.18
map type
data type whose values are tables mapping values of one type (the domain type) to values of another type (the range type)

NOTE Maps are a kind of generalized array. Unlike arrays, the domain and range types may be arbitrary, and possibly infinite types.

4.1.19
entity type
class of objects that may have attributes associated with them

4.1.20
interface-role model
extended Unified Modeling Language class diagram showing the roles and interfaces associated with a logical component, and their mutual relations

4.1.21
logical component instance
incarnation of a logical component: a configuration of objects displaying the behavior defined by the logical component

4.1.22
provides interface
interface that is provided by a role or role instance

4.1.23
requires interface
interface that is used by a role or role instance

4.1.24
specialization
behavioral inheritance
definition, by a role, of behavior which implies the behavior defined by another role

NOTE A role S specializes a role R if the behavior defined by S implies the behavior defined by R, i.e. if S has more specific behavior than R.

4.1.25
diversity
set of all parameters that can be set at instantiation time of a logical component and that will not change during the lifetime of the logical component

4.1.26
mandatory interface
provides interface of a role that should be implemented by each instance of the role

4.1.27
optional interface
provides interface of a role that need not be implemented by each instance of the role

4.1.28
configurable item
parameter that can be set at instantiation time of a logical component, usually represented by a role attribute

4.1.29
diversity attribute
role attribute that represents a configurable item

4.1.30
instantiation
process of creating an instance (an incarnation) of a role or logical component

4.1.31
initial state
state of a role instance or logical component instance immediately after its instantiation

4.1.32
observable behavior
behavior that can be observed at the external software and streaming interfaces of a logical component

4.1.33
function behavior
behavior of the functions in the provides interfaces of a role

4.1.34
streaming behavior
input-output behavior of the streams associated with a role

4.1.35
active behavior
autonomous behavior that is visible at the provides and requires interfaces of a role

4.1.36
instantiation behavior
behavior of a role at instantiation time of a logical component

4.1.37
independent attribute
attribute whose value may be defined or changed independently of other attributes and entities

4.1.38
dependent attribute
attribute whose value is a function of the values of other attributes or entities

4.1.39
invariant
assertion about a role or logical component that is always true from an external observer's point of view

NOTE In reality, the assertion may temporarily be violated.

4.1.40
callback interface
interface provided by a client of a logical component whose functions are called by the logical component

NOTE A notification interface is an example of this, but there may be other call-back interfaces as well, e.g. associated with plug-ins.

4.1.41
callback-compliance
general constraint that the functions in a callback interface should not interfere with the behavior of the caller in an undesirable way, such as by blocking the caller or by delaying it too long

4.1.42
event notification
act of reporting the occurrence of events to 'interested' objects

4.1.43
event subscription
act of recording the types of event that should be notified to objects

4.1.44
cookie
special integer value that is used to identify an event subscription

NOTE Clients pass cookies to a logical component when subscribing to events. Logical components pass cookies back to clients when notifying the occurrence of the events.

4.1.45
event-action table
table associating events that can occur to actions that will be performed in reaction to the events

NOTE This is used to specify event-driven behavior.

4.1.46
non-standard event notification
event notification that is accompanied by other actions (such as state changes of the notifying logical component)

4.1.47
client role
role modeling the users of a logical component

4.1.48
actor role
role (usually a client role) whose active behavior consists of calling functions in interfaces without any a priori constraints on when these calls will occur

4.1.49
control interface
interface provided by a logical component that allows the logical component’s functionality to be controlled by a client

4.1.50
notification interface
interface provided by a client of a logical component that is used by the logical component to report the occurrence of events to the client

4.1.51
specialized interface
interface of a role R that is inherited from another role and is further constrained by R

4.1.52
precondition
assertion that should be true immediately before a function is called

4.1.53
action clause
part of an extended precondition and postcondition specification defining the abstract action performed by a function

NOTE The abstract action usually defines which variables are modified and/or which out-calls are made by the function.

4.1.54
out-call
out-going function call of an object on an interface of another object

4.1.55
postcondition
assertion that will be true immediately after a function has been called

4.1.56
asynchronous function
function with a delayed effect

NOTE The effect of the function will occur some time after returning from the function call.

4.2 Realization terms and definitions

4.2.1
appliance
product as seen by the customer, consisting of the device, an operating system (OS), components and applications

4.2.2
application
software entity that provides a set of functions to a user

4.2.3
component
unit of trading that conforms to the M3W component model

NOTE In the M3W component model, components are containers for models.

4.2.4
component model
specification of what constitutes a component, defining *inter alia* the interaction mechanisms between components and between components and their environment

4.2.5
device
physical part of the appliance, sometimes also used to denote an identifiable element of that appliance

4.2.6
executable component
model that is executable on a platform

NOTE The executable component is a container for services.

4.2.7
M3W system
system that conforms to the M3W specification

4.2.8
operational
state when an M3W system is fulfilling its required functions for a period of time

4.2.9
platform
operating system (OS) and hardware that executes the OS

4.2.10
system
combination of a platform, a set of executable components, a run-time environment and a set of applications that can provide a user with a set of functions

NOTE A platform provides access to the underlying hardware, executable components contain services that provide advanced domain logic, and applications use the logic in the services to provide functions to a user.

5 M3W architecture

5.1 General

This clause gives an overview of the M3W software architecture. In subclause 5.2 we give an overview of the high level software structure of an M3W system, and the separation between the specification of the M3W API and its realization. Subclause 5.3 gives an overview of the specification of the M3W API. Subclause 5.4 gives an overview of the realization technology that shall be used to realize M3W.

5.2 Context

5.2.1 System overview

In an M3W system there are 3 distinct layers:

— Applications: The type of applications will depend on the appliance.

— Middleware layer: This layer consists of M3W and other middleware. M3W can be separated into two parts:

  — Functional part: This provides applications and other middleware with a multimedia platform.

  — Non-functional part: This provides a means to manage lifetime of, and interaction with, realization entities. Furthermore this enables management of non-functional properties, e.g. resource management, fault management and integrity management;

— Computing platform: The platform will depend on the appliance. The API of the computing platform is outside of the scope of M3W. Different realizations of M3W for different platforms will use the different API's for these platforms.

The layers mentioned above are not strict in the sense that applications and other middleware cannot use the computing platform directly. However, the latter is not recommended. The software structure of an M3W system is shown in Figure 1 — Software structure of an M3W system.

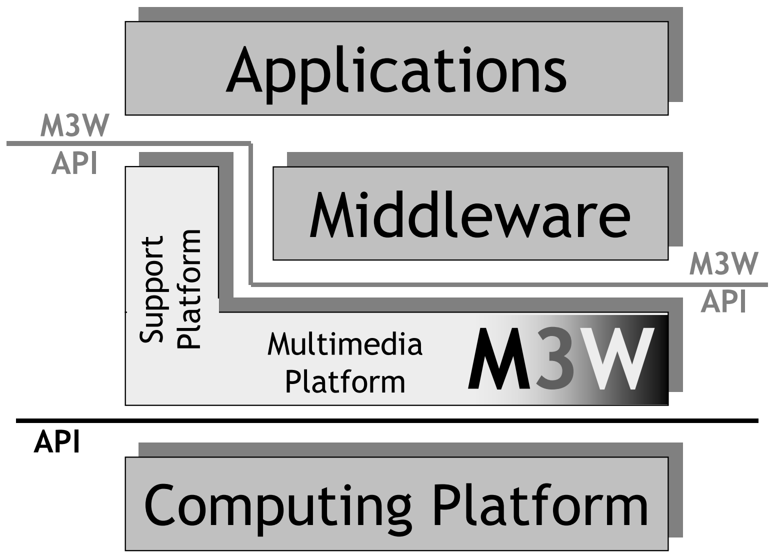

Figure 1 — Software structure of an M3W system

The scope of this standard is limited to the specification of the M3W API and the specification of the realization technology for M3W.

5.2.2 API specification versus realization

In subclause 5.2.1 we introduced the difference between the specification of the M3W API and the realization (technology) that provides an implementation for M3W. The specification of the M3W API provides an uniform view for the applications and other middleware. This uniform view enables the development of applications and other middleware independent of a specific realization of M3W.

The ability to create multiple different realizations for the M3W API enables vendors of M3W (or parts of M3W) to differentiate themselves from competitors.

In order to guarantee the interoperability of multiple parts of M3W provided by different parties, the same realization technology shall be used. Figure 2 — Specification and realization of M3W, illustrates the difference between:

— specification of the M3W API in terms of logical components and roles (see Annex A); and

— realization elements.

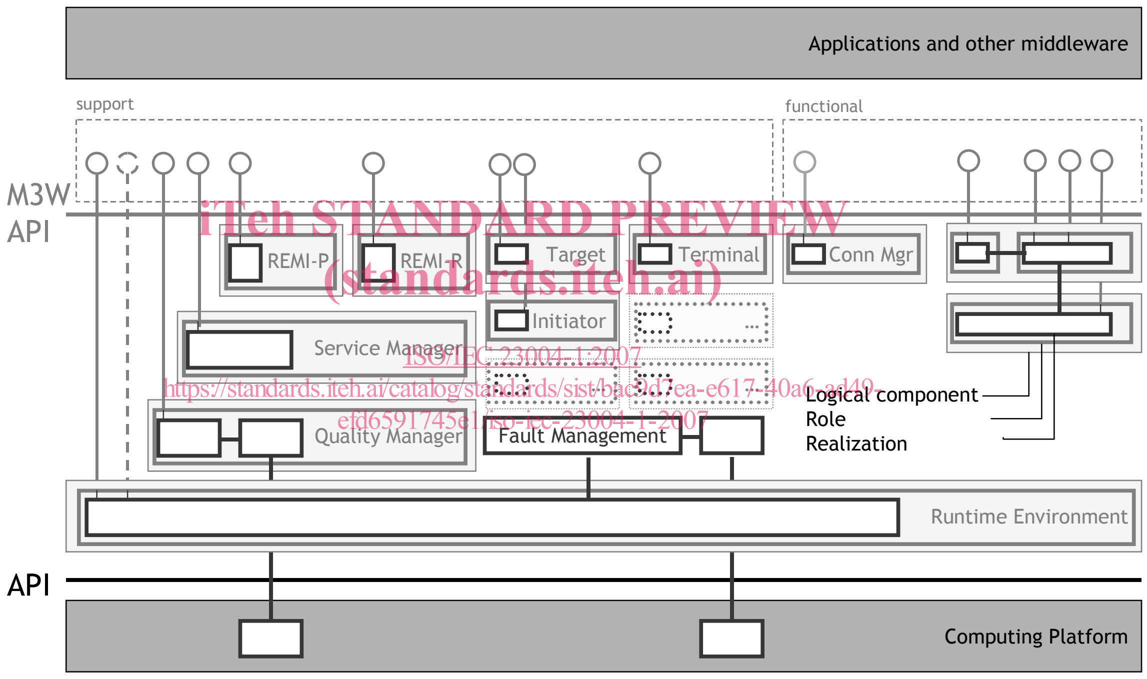

Figure 2 — Specification and realization of M3W

NOTE Figure 2 — Specification and realization of M3W, shows an example realization. Other realizations are possible as long as they comply with the M3W API specification, and the realization technology

5.3 API specification

The specification of the M3W API gives a uniform view to an M3W system. This view is described in terms of logical components and roles (see Annex A), and it should be noted that this view does not contain any realization elements. The M3W API specification consists of the specification of logical components and roles. These logical components and roles give a uniform view to the functionality independent of the realization of this functionality. These logical components and roles can be divided as follows:

— Support:

— Runtime Environment: enables creation of realization elements;