INTERNATIONAL STANDARD

ISO/IEC 23004-3

First edition
2007-09-15

Information technology — Multimedia Middleware —

Part 3: Component model

Technologies de l'information — Intergiciel multimédia —

Partie 3: Modèle de composant

Reference number
ISO/IEC 23004-3:2007(E)

Contents

Page

Foreword iv
Introduction v
1 Scope 1
2 Organization of this document 1
3 Normative references 2
4 Terms and definitions 2
5 Overview of interface suites 3
6 General support interface suites 4
6.1 Instantiation of realization elements 4
6.2 Interaction with and between realization elements 51
7 Overview of realization 108
7.1 General 108
7.2 Guiding principles 108
7.3 Assumptions 109
7.4 Architecture overview 110
7.5 M3W stakeholders 111
7.6 Technical context 116
8 Development framework 118
8.1 Overview 118
8.2 Concepts 118
8.3 Behaviour 127
9 Iterator idiom 129
9.1 Overview 129
10 Execution framework 130
10.1 Concepts 130
10.2 Behaviour 149
11 Service Manager 154
11.1 Description 154
11.2 Service Manager's APIs 154
11.3 Logical component and service 155
11.4 Hierarchical structure of service 156
11.5 Service Manager usages 157
11.6 Metadata 162
12 Remote Method Invoker 191
12.1 Overview 191
12.2 Proxies and wrappers 193
12.3 Marshalling parameters 196
12.4 REMI-R logical component 199
12.5 REMI-P logical component 200
12.6 Remotable service development support 201
Annex A (normative) Service Manager interface definition 203
Annex B (normative) Logical Service metadata schema 205
Annex C (normative) Service metadata schema 207
Bibliography 211

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 23004-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 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 JTC 1/SC 29/WG 11 6981).

The objectives of Multimedia middleware (M3W) are to allow applications to execute multimedia functions with a minimum knowledge of the middleware and to allow applications to trigger updates to the middleware to extend the middleware API. The first goal can be achieved by standardizing the API that the middleware offers. The second goal is much more challenging, as it requires mechanisms to manage the middleware API and to ensure that this functions according to application needs. The second goal can support the first, by reducing the needed standard API to those that provide middleware management. Consequently, applications can use these standard management APIs to generate the multimedia system they require.

ISO/IEC 23004 provides the following:

1) a *vision* for a multimedia middleware API framework to enable the transparent and augmented use of multimedia resources across a wide range of networks and devices to meet the needs of all Users;

2) 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;

3) 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 3: Component model

1 Scope

This part of ISO/IEC 23004 defines the Multimedia Middleware (M3W) Component Model and Core Framework. The context of the M3W Component Model and Core Framework is described in ISO/IEC 23004-1.

2 Organization of this document

This part of ISO/IEC 23004 has the following high level structure:

— Clause 1 defines the scope of this part of ISO/IEC 23004.

— Clause 3 gives an overview of documents that are indispensable for the application of this part of ISO/IEC 23004.

— Clause 4 gives the terms and definitions used in this part of ISO/IEC 23004.

— Clause 5 gives an overview of the interface suites that are part of the Core Framework.

— Clause 6 contains the detailed specification of the interfaces of the Core Framework that are part of the M3W API. These interfaces are structured as follows:

  — Instantiation of realization elements: This subclause contains the interface specifications for instantiation of realization elements based on the uuid of a Service (Run Time) as well as based on the uuid of a logical component (Service Manager).

  — Interaction with and between realization elements: This subclause contains the interface specifications for invocation of an operation on a remote Service as well as enabling remote entities to invoke an operation on a local Service.

— Clause 7 gives an overview of the realization of the component model and core framework.

— Clause 8 describes the development and packaging of realization elements.

— Clause 9 describes the iterator idiom that is used in the execution framework.

— Clause 10 describes the M3W execution framework. This contains the concepts of the component model that are relevant at runtime as well as the elements of the core framework that are needed at runtime for instantiation.

— Clause 11 describes an optional service that enables the instantiation and binding of realization elements based on the uuid of a logical component.

— Service, Meta Data, and Composition: Informative clause that describes the Meta Data of Services and logical components that enables the Service Manager to instantiate and bind realization elements (services) based on the uuid of a logical component.

— Clause 12 describes the optional services that enable the remote invocation of operations of a service.

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 23004-1*, Information technology — Multimedia Middleware — Part 1: Architecture*

W3C REC-xml-20001006, Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000.

W3C REC-xmlschema-1-20041028, XML Schema Part 1: Structures Second Edition, W3C Recommendation 28 October 2004

W3C REC-xmlschema-2-20041028, XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 23004-1 apply.

5 Overview of interface suites

This clause is informative and gives an overview of the interface suites defined in this International Standard. The interface suites defined in this document belong to the core component model and support framework that is specified in M3W. The interface suites that belong to the core component model and support framework deal with instantiation of realization elements and interaction with (and between) realization elements.

Subclause 6.1 specifies the interfaces suites for Instantiation of realization elements. This contains the specification of the Runtime Environment and Service Manager role. The Runtime Environment logical component enables the instantiation of realization element based on an UUID of the realization element (service). The Service Manager logical component enables the instantiation and binding of a number of realization elements that realize a certain logical component (based on the UUID of the logical component).

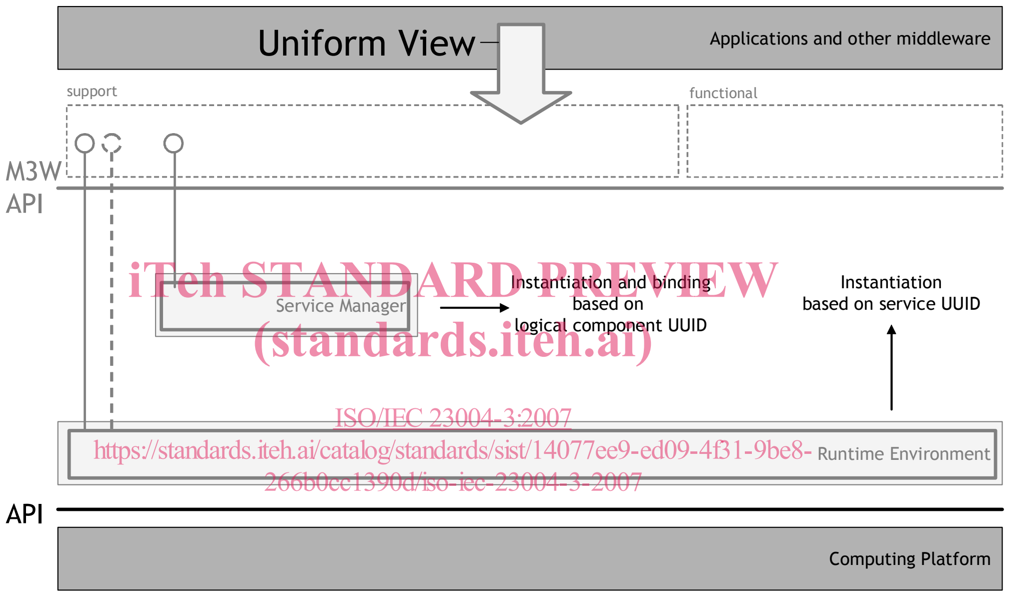
Uniform View
Applications and other middleware
support
functional
M3W
API
Service Manager
Instantiation and binding based on logical component UUID
Instantiation based on service UUID
Runtime Environment
API
Computing Platform

Figure 1 — Overview of the interface suites (logical components) for creation of realization elements

Subclause 6.2 specifies the interface suites for remote method invocation. This contains the specification of REMI-P used for providing operations that can be invoked from a remote M3W device. It also contains the specification of REMI-R used for invocation of operations provided by services on a remote M3W device. A realization of REMI-P must be available on the M3W device that hosts the service that is invoked. A realization of REMI-R needs to be available on the M3W device that has the invoking entity.

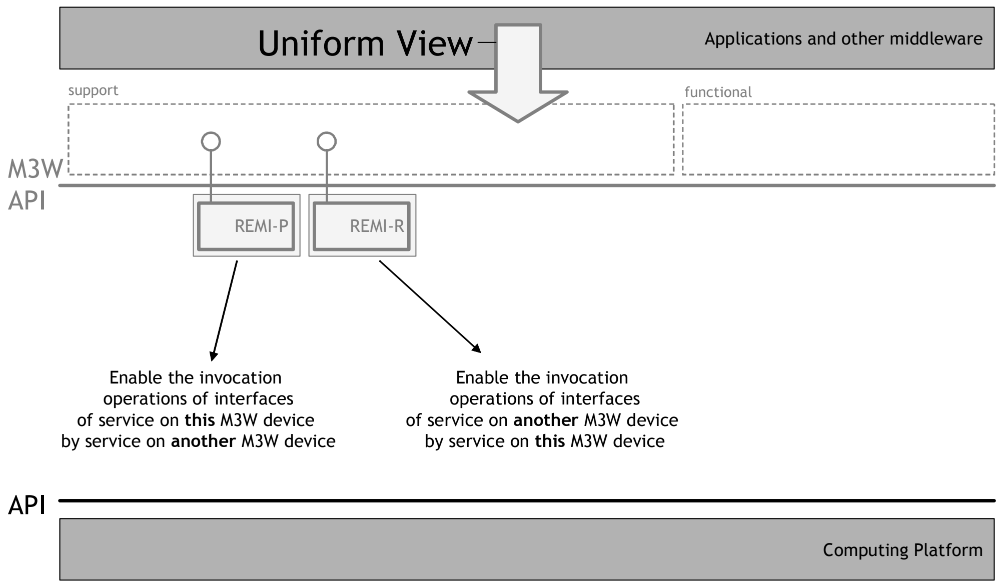
Uniform View
Applications and other middleware
support
functional
M3W API
REMI-P
REMI-R
Enable the invocation operations of interfaces of service on this M3W device by service on another M3W device
Enable the invocation operations of interfaces of service on another M3W device by service on this M3W device
API
Computing Platform

Figure 2 — Overview of interface suites (logical components) for remote method invocation

6 General support interface suites

6.1 Instantiation of realization elements

6.1.1 Runtime Environment

6.1.1.1 Concepts

The function of a Runtime Environment logical component is to enable the instantiation of realization elements (Services, see 10.1.7). It can instantiate Services on request of a client. Furthermore the Runtime Environment offers a number of interfaces for the inspection and manipulation of the Registry that contains information on the executable components (see 10.1.14) that contain the Services that can be instantiated.

6.1.1.2 Types & Constants

6.1.1.2.1 Public Types & Constants

6.1.1.2.1.1 Error Codes

Signature

```
const rcResult RC_ERR_RUNTIME_NOT_IMPLEMENTED = 0x00000001
const rcResult RC_ERR_RUNTIME_CANNOT_INITIALIZE = 0x00000002
const rcResult RC_ERR_RUNTIME_CANNOT_FINALIZAE = 0x00000004
const rcResult RC_ERR_RUNTIME_NO_SUCH_ELEMENT = 0x00000008
const rcResult RC_ERR_RUNTIME_NO_SUCH_SERVICE = 0x00000010
const rcResult RC_ERR_RUNTIME_INVALID_LOCATION = 0x00000020
const rcResult RC_ERR_RUNTIME_INVALID_COMPONENT = 0x00000040
const rcResult RC_ERR_RUNTIME_ALREADY_REGISTERED = 0x00000080
const rcResult RC_ERR_RUNTIME_UNKNOWN_COMPONENT = 0x00000100
const rcResult RC_ERR_RUNTIME_NOT_ALLOWED = 0x00000200
```

Qualifiers

— Error-codes

Description

The non-standard error codes that can be returned by functions of this logical component

Constants

Table 1

Name	Description
RC_ERR_RUNTIME_NOT_IMPLEMENTED	This error is returned when a particular function is not implemented by the runtime environment
RC_ERR_RUNTIME_CANNOT_INITIALIZE	This error is returned when a component containing a service cannot be initialized
RC_ERR_RUNTIME_CANNOT_FINALIZE	This error is returned when an executable component cannot be finalized.
RC_ERR_RUNTIME_NO_SUCH_ELEMENT	This error is returned when an (instance of an) element is requested that is not available
RC_ERR_RUNTIME_NO_SUCH_SERVICE	This error is returned when a service instance is requested and the service is not available
RC_ERR_RUNTIME_INVALID_LOCATION	This error is returned on an attempt to use an invalid location. For example for registration of an executable component
RC_ERR_RUNTIME_INVALID_COMPONENT	This error is returned on an attempt to register an invalid executable component.
RC_ERR_RUNTIME_ALREADY_REGISTERED	This error is returned on an attempt to register an executable component multiple times
RC_ERR_RUNTIME_UNKNOWN_COMPONENT	This error is returned on an attempt to modify the registered information related to an executable component that is not known to the runtime environment
RC_ERR_RUNTIME_NOT_ALLOWED	This error is returned when the operation is not allowed

6.1.1.2.1.2 rcRuntime_ComponentRecord_t

Signature

```
struct _rcRuntime_ComponentRecord_t {
           UUID cmpId;
           String location;
} rcRuntime_ComponentRecord_t, *prcRuntime_ComponentRecord_t;
```

Qualifiers

— struct-element

Description

Data structure used to pass and store registration of an executable component.

6.1.1.2.1.3 rcRuntime_ContainerRecord_t

Signature

```
struct _rcRuntime_ContainerRecord_t {
           UUID cmpId;
           UUID svcId;
} rcRuntime_ContainerRecord_t, *prcRuntime_ContainerRecord_t;
```

Qualifiers

— struct-element

Description

Data structure used to pass and store registration of the containment of a service by an executable component.

6.1.1.2.1.4 rcRuntime_CompliesRecord_t

Signature

```
struct _rcRuntime_CompliesRecord_t {
           UUID complying;
           UUID blueprint;
} rcRuntime_CompliesRecord_t, *prcRuntime_CompliesRecord_t;
```

Qualifiers

— struct-element

Description

Data structure used to pass and store registration of the complies relation between services.

6.1.1.2.2 Model Types & Constants

None

6.1.1.3 Logical Component

6.1.1.3.1 Interface-Role Model

Figure 3 — Interface-Role Model, depicts the interface-role model of the Runtime Environment (grey box). It shows the interfaces and the roles involved with this component.

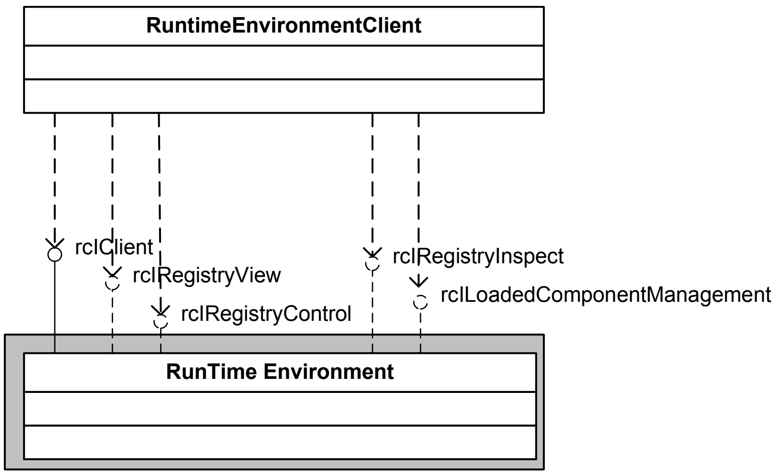

Figure 3 — Interface-Role Model

A Runtime Environment Logical Component contains the Runtime Environment role that provides the rcIClient interface. This interface can be used by a client to create instances of services. The Runtime Environment role also has a number of optional interfaces to modify and inspect the registry of the Runtime Environment.

6.1.1.3.2 Diversity

6.1.1.3.2.1 Provided Interfaces

Table 2

Role	Interface	Presence
Runtime Environment	`rcIClient`	`Mandatory`
Runtime Environment	`rcIRegistryView`	`Optional`
Runtime Environment	`rcIRegistryControl`	`Optional`
Runtime Environment	`rcIRegistryInspect`	`Optional`
Runtime Environment	`rcILoadedComponentManagement`	`Optional`

6.1.1.3.2.2 Configurable Items

The behaviour of the Runtime Environment depends on the registry. This registry contains the information the available executable components, which services are contained by these executable components and which services are compatible. The registry contents is specified using the following attributes.

Table 3

Role	Attribute
Runtime Environment	`componentRecords`
Runtime Environment	`containerRecords`
Runtime Environment	`compliesRecords`

6.1.1.3.2.3 Constraints

None

6.1.1.3.3 Instantiation

6.1.1.3.3.1 Objects Created

The logical component Runtime Environment and the Runtime Environment role are always available. They are not created by the clients of the M3W.

Table 4

Type	Object	Multiplicity
`RuntimeEnvironment`	`runtime`	1

6.1.1.3.3.2 Initial State

The following constraints apply to the initial state of a logical component instance:

— none

6.1.1.3.4 Execution Constraints

The logical component Runtime Environment is thread-safe.

6.1.1.4 Roles

6.1.1.4.1 Runtime Environment

Signature

```
role RuntimeEnvironment {
  rcRuntime_ComponentRecord_t       componentRecords[];
  rcRuntime_ContainerRecord_t       containerRecords[];
  rcRuntime_CompliesRecord_t        compliesRecords[];
}
```

Qualifiers

— root

Description

A Runtime Environment enables the instantiation of service based on the registry contents (`componentRecords`, `containerRecords` and `compliesRecords`). Furthermore it provides interfaces for the inspection and manipulation of this registry.

Independent Attributes

Table 5

Name	Description
`componentRecords`	`Used to store the uuid's of the registered executable components and the location on locally accessible storage.`
`containerRecords`	`Used to store the binding between services and executable components`
`compliesRecords`	`Used to store the complies relation between services.`

Invariants

— When the `containerRecords` contain the association between a service and an executable component then `componentRecords` contains the association between this executable component and a location on locally accessible storage.

Instantiation

The Runtime Environment role is always instantiated as part of the Runtime Environment logical component.

Active Behaviour

The Runtime Environment has no active behaviour

6.1.1.5 Interfaces

6.1.1.5.1 rclClient

Qualifiers

None.

Description

This interface of a Runtime Environment role provides facilities for instantiation of realization elements (services).

Interface ID

uuid(41c235af-0417-4731-8627-c11c3d51d359)