INTERNATIONAL STANDARD

ISO/IEC 23004-4

First edition
2007-09-15

Information technology — Multimedia Middleware —

Part 4: Resource and quality management

Technologies de l'information — Intergiciel multimédia —

Partie 4: Management des ressources et de la qualité

Reference number
ISO/IEC 23004-4:2007(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword 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 2
5.1 Introduction 2
5.2 Resource management 2
5.3 Quality management 3

6 Resource management interface suite 6
6.1 Overview 6
6.2 Types and constants 6
6.3 Logical component 20
6.4 Roles 20
6.5 Interfaces 21

7 Overview of realization 29

8 Resource management 30
8.1 Overview 30
8.2 Responsibilities of roles in the Resource Management Framework 30
8.3 Realization of a resource chief 31
8.4 Composition of quality information 34

Annex A (informative) **Dynamic view of the Resource and Power Management Framework** 36

Annex B (informative) **CPU chief details** 38

Annex C (informative) **Approach to power management** 40

Annex D (informative) **Composition of quality information example** 43

Bibliography 47

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 23004-4 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 N 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 APIs to those that provide middleware management. Consequently, applications can use these standard management APIs to generate the multimedia system they require.

The aim of M3W is to define a component-based middleware layer in high-volume embedded appliances that enables robust and reliable operation. These types of product are heavily resource constrained, with a high pressure on silicon cost and power consumption. In order to be able to compete with dedicated hardware solutions, the available resources will have to be used very cost-effectively, while enabling robustness and meeting stringent timing requirements imposed by high-quality elements such as digital audio and video processing.

High-volume embedded appliances are considered business-critical devices. Their failure can have important economic implications for the producing company. Hence, they have stringent and demanding quality requirements. Unfortunately, in the software industry misbehaving products are commonplace. However, this is not the case with consumer electronics, home appliances and mobile devices. Users are accustomed to robust and well-behaved devices.

In consumer electronics, there are demanding quality requirements and a need to use the available hardware resources cost-effectively. The term “Application” is used to refer to the software entity that (indirectly) provides certain functionality to an end-user; the Application may include Service Instances that are bound to it. In order to ensure that an Application provides the correct service, it needs to be assigned the hardware resources it requires. This can be achieved by assigning it the “worst case” resources required. However, with many applications, the worst-case resource usage is much less than the mean case. If this worst-case approach is followed, hardware resources will be wasted. On the other hand, problems may arise if the running applications require more resources than can be made available. If the platform follows a typical fair policy for the assignment, the behaviour of the system will be unpredictable, especially for applications that must provide results according to some specified time constraints.

The goal of Resource Management is to assign budgets or resource reserves to Applications. These budgets are guaranteed, so that they are available in any situation. This approach helps to enhance system robustness, because an Application is unable to affect the resource reserves of others.

In order to benefit from Resource Management, Applications participating in Resource Management should be resource-aware (RA). This means that they are aware of the resources they need during their execution and should adapt their behaviour to the resources which are made available. This ensures that applications can function correctly without exhausting all of the system resources that have been allocated to them.

Quality-Aware (QA) Applications are RA applications that are aware of the quality of service that they deliver. Typically, they are capable of providing different quality levels. They are characterized by the quality they provide and the resources needed for this purpose. Usually the higher the quality level, the higher the resource needs. This type of Application is able to dynamically change the provided quality level, depending on the assigned budgets.

Often, QA and RA applications are real-time applications. The assigned budgets allow them to provide a suitable output within some time interval. A hard real-time application can be viewed as QA with only two quality levels: maximum quality or nothing. The Resource Management framework can also deal with Non Resource-Aware Applications (NRA) in the sense that they are assigned a fixed budget all together.

The relation between the Resource Management framework and QA applications is based on a **contract model**. The system provides resources and the Applications commit to generate the required results (outputs) with a specific and stable quality. Budget assignment must be obtainable, which means that it should not be possible to assign more resources than those actually available. Contracts are negotiated with the Applications with the goal of maximizing overall system quality, as perceived by the user. In this process, the "importance" of the Applications is a primary parameter for this operation.

Power is considered to be one of the most important resources to be managed in the next generation of consumer electronics. Its importance is clear for mobile devices, where the goal is to maximize battery life. In stationary devices, it is also relevant for environmental conditions, fan-less operation and to increase the lifetime of the silicon devices. In addition, power management is related to heat control, so that the temperature of different parts of the device can be maintained within a certain threshold.

The basic goals in M3W with respect to power management are as follows.

— Reduce power consumption.

— Increase battery life, for mobile devices.

— Ensure a system-wide limit for heating: If it is detected that the temperature of the device it is too high, it may be required to reduce it.

— Take advantage of power-aware hardware and M3W components.

— Ensure that the system and relevant Applications are always executed, in spite of the power saving policy selected.

Power management is very much related to resource management. A number of techniques for reducing power consumption are based on moving hardware components to less power-consuming modes of operation, which implies reducing/modifying the available resources (CPU capacity, memory size, bandwidth, etc.). This is the case, for example, with CPU voltage and frequency scaling, which can reduce power consumption and consequentially, CPU computational power. For this reason, it is desirable to integrate power management with quality and resource management.

The Resource Management Framework is in charge of achieving the functionality that has been sketched in this introductory clause. Its functions and architecture are defined in this part of ISO/IEC 23004.

Information technology — Multimedia Middleware —

Part 4:
Resource and quality management

1 Scope

This part of ISO/IEC 23004 defines the Resource and Quality Management framework of the MPEG Multimedia Middleware (M3W) technology.

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 provides an overview of and introduction to the Resource and Power Management Framework.

— Clause 6 contains a detailed specification of the interfaces of the Resource and Power Management Framework that are part of the M3W API.

— Clause 7 gives an overview of the realization of the Resource and Power Management Framework.

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

Annex A, Dynamic view of the Resource and Power Management Framework, describes the interactions between the different entities of the Resource and Power Management Framework.

Annex B, CPU chief details: The Resource and Power Management Framework distinguishes four different roles that are together responsible for Resource and Power Management:

1) resource chief;

2) resource manager;

3) quality chief;

4) quality manager.

The CPU chief is discussed in this annex. The CPU chief is a specialization of a resource chief. This is the chief responsible for monitoring and enforcing the CPU budgets.

Annex C, Approach to power management: Power is a resource as well. This annex discusses managing this specific resource.

Annex D, Composition of quality information example: Often we need to manage the Resource Consumption and delivered Quality of a composition of entities instead of a single atomic entity. This annex discusses how to compose the Quality and Resource information in such cases.

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*

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

5.1 Introduction

This is an informative clause that gives an overview of the API specification of the resource management framework.

5.2 Resource management

The optional M3W Resource Management Framework provides timely, guaranteed and protected access to system resources. These functions allow Applications to specify their resource demands, leaving the Resource Management Framework to satisfy those demands using multimedia platform specific resource management schemes. In this situation, two models (or views) can be defined:

— Resource specification models: Applications and Services must specify the budget or resource share they need. For this purpose, the runtime provides a certain model and an associated interface. If it is feasible, a given resource share can be committed to these entities.

— Resource management models: The runtime must provide a means for accounting, enforcing and monitoring resource usage. In this way, it is possible for the runtime to guarantee the committed budgets and to provide useful information to applications so that they can adapt their behaviour to their current resource shares / budgets.

These two models are somewhat independent. For a given specification model, a number of management models can be implemented to satisfy it, and vice versa.

The resource specification model is based on assigning resource shares or budgets to the Applications. As an example, the specification model for handling memory and CPU is now described:

— Memory: The corresponding entities should make a claim for a certain amount of memory. The manager should respond whether the grant can be approved or not. Ideally, these entities should request all the memory they will need for their entire execution lifetime. If they dynamically claim memory, it could be that in the middle of its execution there is no additional free memory, and the program is then unable to continue with its execution. On the other hand, it may be difficult for an application to know in advance the exact memory needed, and it may ask for much more than it really needs. In any case, memory feasibility tests are very simple and can be made frequently, with little overhead.

— CPU: The specifying model is based on requesting a certain CPU share (percentage) for a given period of time. This budget is refilled periodically. CPU budgets can either be allocated to a thread or to a group of threads (also named a cluster). For example, a thread may be allowed to use the CPU for 1ms every 10ms (budget is 1 ms and the refill period is 10ms). In the context of this specification, a thread is the unit of concurrency.

Resource shares are assigned by the Resource Management framework and shall be enforced during system operation. For this purpose, it is necessary to account for resource usage and take some action if a budget user tries to use more resources than budgeted. If an Application requests more resources, it should be checked whether this is possible, given the current system situation. If there are available resources, the request may be satisfied. Otherwise, it should be denied or a new system assignment should be made.

QA Applications and Services have static information on their quality levels and their associated resource needs. In the case of the CPU, the required budget is an estimation that is e.g. calculated by measurements taken when running the program with a meaningful set of input streams and data. In some domains, e.g. multimedia, the real resource needs depend on the type of data, so it is difficult to provide accurate estimations. Applications may dynamically adapt and update their required budget estimations for use in future negotiations. The same considerations can be made with respect to memory, although typically in this case the required memory is subject to less variation, and estimations can be made more accurately.

An Application or Service may provide the required quality level with different sets of resources. For example, it can use a lot of CPU and little memory or only a little CPU but with plenty of memory to deliver the same quality level. In the context of the Resource Management Framework the set of resources that an Application requires for a certain quality level is called a configuration. A quality level can have several configurations associated. The Resource Management framework is in charge of deciding which configuration will be selected, depending on the needs of the applications and the available resources in the system.

In a M3W system, Resource Aware and Non-Resource Aware applications can co-exist. The approach for dealing with this situation is to divide the system resources into two partitions:

— RA: This partition is composed by all Resource Aware Applications and Services. The QM has assigned them a certain budget, which shall be guaranteed, for the execution (until a re-negotiation).

— NRA: This partition includes the rest of resources and it is used for the execution of the Non-Resource Aware Applications and Services.

One possible way of implementing this division is by assigning priorities to RA threads (those that belong to RA Applications or RA Services) that are higher than those assigned to NRA elements. When the budget of a RA element is exhausted, its priority could be reduced. When the budget is replenished, the priority is raised. A real-time operating system, with appropriate scheduling policies, is required for the execution of the RA threads. The scheduling of NRA threads can be done using a classical quantum scheduler.

Power management is an important aspect of resource management, as the available system resources depends on the power settings of the corresponding hardware components. If the power settings are changed, (for example to reduce consumption, then the negotiated resource shares cannot be kept, and the contract may be broken. For this reason, the approach is to consider power as another resource, although with special characteristics, and it is managed together with the rest of the system resources. More details on power management can be found in Annex C.

5.3 Quality management

The purpose of quality management is to drive the interaction with the QA Applications, so that the quality of the overall system is maximised. A system may be composed of a set of QA Applications. Where each application can provide a set of quality levels. A quality level is characterized by a description of the provided quality and by the required resources for this purpose. The problem is to decide the quality level for each Application, and as a consequence the resource assignment, so that system quality is maximized and the resource assignment is feasible.

The criteria for making this selection vary from system to system. The information that has to be considered includes:

— Importance of Applications: There are applications that are fundamental in a particular device and should always provide some minimal quality, such as the handling of incoming calls in a mobile phone. Information from the execution context is also useful for identifying the important Applications. In a system with multiple windows, it is usually the case that the Application running in the window with the user focus is the most important.

— User settings: The goal is to maximize user satisfaction. Hence, it is important to consider user settings.

— Domain knowledge: This information is used in the decision process, e.g. to decide which Application is most important and has to receive more resources.

The quality management functions do not finish when a contract is negotiated. It is necessary to monitor the system to handle situations where the system is overloaded or has free resources. The resources required by Applications are usually estimations. In multimedia applications, the required resources strongly depend on the input data. In particular, the computation time may depend on whether the scene is static or with a lot of action and on the compression algorithm used. Hence, it can be that an Application requires more resources or has more than needed. In this situation, there should be mechanisms for its correction. As a consequence, the resources needed for the quality level may change and the contracts must be (at least partially) renegotiated.

The operations that quality management provides can be described as follows:

— Reservation is the most basic mechanism for providing QoS guarantees. It consists of assigning shares of resources (budgets) to Applications and guaranteeing them based on runtime monitoring and enforcement of their usage. Most of the functions related to reservation are performed as part of resource management.

— Negotiation can be defined as the operation for creating a new contract (or modifying the current one) between the system software and the different Applications. This long-term contract determines a set of minimum requirements for both sides. The system guarantees a minimum budget, and the Application agrees on executing the functionality required by the user providing, at least, a minimum utility. The process is based on having a dialogue between both domains to check whether new settings are feasible in the system. Differentiating negotiation (feasibility test) from the actual settings into two different phases is useful when we want to check the feasibility of several mutually exclusive configurations, and then select one of them. Negotiation can be executed at Application (or Service) start-up, or dynamically when it is detected that the current contract is no longer valid. This renegotiation may be in response to either external or internal events. External events are those signalled externally to the system, while internal events are those detected by runtime monitoring.

— Setting is the operation of making the system configuration resulting from a negotiation operational. As part of this procedure it is necessary to communicate to the Applications their executing quality level and to set the resource budgets, by using the facilities provided by the resource manager. This configuration change has to be done following a certain procedure that depends on how abrupt the changes can be made. Some Applications require a smooth change, so it is necessary to follow a protocol that ensures this requirement. For example, Applications that reduce their quality level may have to change before others raise theirs, in order to ensure that budgets are guaranteed during the setting operation.

— Monitoring is the process in charge of obtaining information about the execution of Applications and their threads, in order to know how they behave. This information includes (i) resource usage made by Applications and threads, (ii) progress monitoring (usually in the form of milestones reached or percentage of processing finished), (iii) adaptation measurements (performance achieved by threads, and data dropping made by Applications), and (iv) application domain metrics (utility provided as a function of provided QoS parameters and QoS violations).

— Adaptation is the operation of responding to transient overloads, detected at runtime, by performing some short-term changes. The goal is to adapt the processing algorithm in order to have a near-constant

computational complexity. The main characteristic of adaptation is that the current contract between the system domain and the Application domains must continue to be fulfilled.

— Optimization is the process of achieving the best configuration for all the entities in the system, and hence achieving the best use of available resources at each point in time. Negotiation determines a contract between the system domain and the Application domains that defines the minimum requirements for them, and optimization can make small changes to current configurations but without changing the contract. The dynamic nature of multimedia systems makes that a certain configuration that is optimal for the current state it is no longer optimal when the situation changes. Optimization tries to anticipate the need for resource reallocation, enhancing the medium and long-term behaviour of services. It is closely related to adaptation and negotiation, but the main difference is the mechanism that triggers their execution. Negotiation is executed when the new situation (for example, when the user launches a new application) requires the modification of the current contract between the system domain and the application domains. Optimization is executed periodically by the Resource Management framework's control system to try to make small changes to improve the behaviour of the system (without changing the contract). Finally, adaptation is triggered by transient overload situations detected at runtime.

The QA Applications have to follow a certain procedure for interacting with the Resource Management framework. First of all, they have to be implemented in such a way that the contract is met. They should not try to use more resources than budgeted and have to provide the committed quality. QA Applications have to provide some operations related to the interaction with the QoS management facilities:

— Provide quality information. This includes the quality levels and the resources required for each quality level. This information is basic for the negotiation process. Applications have to update this information whenever it changes. This may be caused by changes in the estimations of resources needed, quality levels that can or cannot be provided, for example due to the nature of the input data, etc.

— Follow the assigned settings. The Application receives a command to set the quality level to be executed after a negotiation. It has to change its execution mode accordingly. Applications also have to notify the completion of the change to the Resource Management framework.

— Notify information and events. It is useful to provide information on the behaviour of the Application to the QoS manager, so that it can use this information to optimize system operation. Events should also be notified to the QoS manager. Some relevant events are fault detection, request for changing the contract, etc.

Quality management includes power management. Information on the current power status and user power settings must be taken into account in the operations specified above, and especially with negotiation. The available resources depend on the selection of the power settings. Hence, it is needed to find a suitable power setting that meets user power requirements and that allows for providing the minimum global quality required by the user.

In addition, it is interesting to be able to take advantage of Services with algorithms that are specifically designed for reducing power consumption. A Quality-Aware Service may have different resource configurations for providing a quality level. Some of these configurations may include power-saving algorithms. In order to include this type of support in the Resource and Power Management Framework, it is required to be able to:

— Include information on the power consumption level of the configurations associated to a quality level.

— Manage the power consumption level information in the quality manager: the quality manager has to be aware of this qualification and take advantage of it. For example, it could select configuration of quality levels that requires low energy, when the system is in a low-power mode.

6 Resource management interface suite

6.1 Overview

In this clause the main types and functions are commented, to facilitate understanding of the API. This description deals initially with the data types and then the logical components are described. Following that, a description of the roles is given. Finally, the Interfaces and their operations are described.

6.2 Types and constants

6.2.1 RcQoSEventId

Signature

```
enum RcQoSEventId
{
    QOS_EVENT_OVERRUN,
    QOS_EVENT_BUDGET_TOO_LOOSE,
    QOS_EVENT_BUDGET_TOO_LOW,
    QOS_EVENT_UNKNOWN,
    MAX_QOS_EVENTS
};
```

Qualifiers

None

Description

Resource management related events.

6.2.2 RcQoSEventTime

Signature

```
typedef Int32 RcQoSEventTime;
```

Qualifiers

None

Description

Time that a event occurred.

6.2.3 RcQoSEventData

Signature

```
typedef pVoid RcQoSEventData;
```

Qualifiers

None

Description

Event specific data.

6.2.4 RcQoSEventInfo

Signature

```
struct RcQoSEventInfo
{
    RcQoSEventTime eventTime;
    RcQoSEventData eventData;
};
```

Qualifiers

None

Description

Information associated with an event.

6.2.5 RcQoSImportance

Signature


```
typedef Int8 RcQoSImportance;
```

Qualifiers

None.

Description

This represents how important the execution of an application is for the user of the device. Important applications are favoured in the resource assignment and the QoS-Manager tries to execute them with a high quality level.

6.2.6 RcQoSApplicationId

Signature

```
typedef Int8 RcQoSApplicationId;
```

Qualifiers

None.

Description

This is the Resource Management Framework application identifier. Its value is provided by the Resource Management framework.

6.2.7 RcQoSResourceId

Signature

```
enum RcQoSResourceId
{
    RID_CPU
    RID_NETWORK_BASE,
    RID_NETWORK_TOKENBUCKET,
    RID_POWER,
    QOS_MAX_NUM_RESOURCES
};
```

Qualifiers

None.

Description

This ID describes the type of resource.

Values

Table 1

Name	Description
• RID_CPU	A 'CPU' resource
• RID_NETWORK_BASE	A 'network base' resource
• RID_NETWORK_TOKENBUCKET	A 'network token bucket' resource
• RID_POWER	A 'power' resource
• QOS_MAX_NUM_RESOURCES	The maximum number of resources

6.2.8 RcQoSDescription

Signature

```
typedef String RcQoSDescription;
```

Qualifiers

None.

Description

Type used to describe entities in the Resource Management and Power Management framework. For example this type is used to describe Quality Levels, as well as components.

6.2.9 RcQoSBudgetId

Signature

```
Int8 RcQoSBudgetId;
```

Qualifiers

None.

Description

This is the identifier of the budget.

6.2.10 RcQoSBudgetUser

Signature

```
typedef pVoid RcQoSBudgetUser;
```

Qualifiers

None.

Description

This is the user of a budget.

6.2.11 RcQoSBudgetInfo

Signature

```
typedef pVoid RcQoSBudgetInfo;
```

Qualifiers

None.

Description

Describes a budget. For example this is used to express required budgets by a logical user for a certain Quality Level.

6.2.12 RcQoSLogicalUserName

Signature

```
typedef String RcQoSLogicalUserName;
```

Qualifiers

None.

Description

String that contains the name of a logical user. Budgets are assigned to logical users. A logical user can consist of a number of budget users.