UNIVERSITAT POMPEU FABRA

This thesis entitled

“Contribution to an Architecture for Multimedia Information Management and Protection Based on Open Standards”

Written by Víctor Torres Padrosa

And directed by Dr. Jaime Delgado Mercé

Has been approved by the Department of Information and Communication Technologies

Date: May 23th, 2008

A thesis submitted in partial fulfillment of the requirements for the Degree of *Doctor per la Universitat Pompeu Fabra*

Dipòsit legal: B.54974-2008
ISBN: 978-84-692-0968-4

Author: Víctor Torres Padrosa

Thesis advisor: Catedràtic d'Universitat Jaime Delgado Mercé

Contribution to an Architecture for Multimedia Information Management and Protection Based on Open Standards

Abstract

The main goal of this research project is to participate in the definition of a generic architecture for the protection and management of digital rights in a multimedia content management scenario following open standards as much as possible.

In order to achieve our goal, several standards and solutions will be analysed and taken into account. A relevant standard to be considered is MPEG-21, as it specifies a multimedia framework that provides interoperability among systems that deliver multimedia content. On the other hand, the requirements of other de facto standards and initiatives will also be analysed so as to define a general architecture that is not only restricted to manage a specific kind of content or format.

The first part of the contribution involves the integration different parts of the MPEG-21 standard: Digital Item Declaration (DID), Rights Expression Language (REL), Rights Data Dictionary (RDD), Digital Item Processing (DIP) and Event reporting (ER).

The second part of the contribution consists in the definition of a generic DRM architecture. Fist, a preliminary architecture for the protection and management of digital rights for multimedia content has been tackled. This preliminary architecture is mainly based on the MPEG-21 standard, as a result of the integration work that had been previously performed. Later on, a more general architecture has been defined. Several implementations are described such as those in several Spanish and European projects, such as AXMEDIS, VISNET II, Linked-Work and GILDDA. The focus in made on use cases, security features and a testing methodology. Linked-Work section details a specific implementation, emphasising on the key concepts that make it different from other kind of DRM systems.

A mis padres, por vuestro impulso.
Siempre estaré en deuda con vosotros.

A la Maria Àngels, pel teu afecte,
comprensió i el teu incansable suport.

Contents

List of Figures

List of Tables

Part I. Introduction

1 Introduction

This introductory chapter serves to three purposes. On one hand, it defines the scope of the work that is addressed by this document. On the other hand, it facilitates the reading of the document by giving and overview of its structure. Finally, but not less important, it provides the necessary elements to enclose the work in the framework of a research project.

1.1 Objectives

The main goal of this research project is to define a generic architecture for the protection and management of digital rights in a multimedia content management scenario following open standards as much as possible. This ambitious and broad aim is included in a general research framework, which is being carried out by the Distributed Multimedia Applications Group (DMAG) [1].

In order to achieve our goal, several standards and solutions need to be analysed and taken into account. A relevant standard to be considered is MPEG-21 [2], as it specifies a multimedia framework that provides interoperability among systems that deliver multimedia content. On the other hand, the requirements of other *de facto* standards and initiatives will be also analysed so as to define a general architecture that is not only restricted to manage a specific kind of content or format.

The DMAG long experience in the MPEG-21 area has been exploited in this sense to develop several new software tools that integrate additional functionalities together with the already existing DMAG software modules in order to show the results and implications of the integration work. The already existing software tools correspond to different research works performed under the same research group framework.

The first milestone to achieve the general goal of defining a generic DRM architecture involves working in the integration of different parts of the MPEG-21 standard. The obtained results will help to understand the operation of a multimedia system and to identify the requirements of an architecture that is able to manage such kind of multimedia information, which is the main goal of this research work.

The second milestone in this sense consists in going one step further and defining the functionality and operation of a MPEG-21 based preliminary DRM architecture. This preliminary architecture will be mainly based on the MPEG-21 standard, as a result of the integration work achieved in the previous milestone.

The third milestone is the definition and specification of a generic DRM architecture, which will be called Multimedia Information Protection and Management System (MIPAMS), taking into account not only the knowledge from the previous work in MPEG-21, but also the requirements of other initiatives and the relevant features present in other existing commercial or research systems. In this sense, the analysis presented in the State of the Art will be very relevant to the research work.

The fourth milestone corresponds to the validation of the proposal in different contexts such as European and National projects, where an implementation will be provided and a testing methodology will be proposed.

Finally, the fifth milestone will consist in further investigating the needs and implications of DRM systems in a context more focused on the content creation and derivation. The MIPAMS architecture will be used as the basis of this work, but we will investigate whether some advanced functionality that is not usually present in a DRM architecture is needed.

1.2 Structure of the document

This document is mainly composed of two parts, i.e. the “State of the Art” and the “Contribution”, as well as other smaller sections such as the introduction or the conclusions and future work.

The State of the Art part of the document is included in a single chapter (chapter 2) and tries to compile in a coherent way all the results of the exploratory stage. All the concepts related to the project are defined and described, including the difficult task to weave all the relationships between them. In this part we present the different parts of the MPEG-21 standard related to this research work: Digital Item Declaration (DID) [3], Rights Expression Language (REL) [4], Rights Data dictionary (RDD) [5] and Digital Item Processing (DIP) [6]. Moreover, we analyse several DRM standards and initiatives whose features are relevant for the definition of the DRM generic architecture. Some of the selected systems are open source, whereas some others have open specifications or give some details about their implementation or usage. From the wide range of existing systems, we have selected the Open Mobile Alliance Digital Rights Management (OMA DRM) [7], Sun’s DReaM [8], Marlin [9] [10], Open Secure Digital Rights Management (OpenSDRM) [11], Windows Media Rights Manager [12] [13], Apple iTunes [14] , CORAL [15] [16] [17] and the Digital Media Project (DMP) [18].

The Contribution part of the document describes the research that has been carried out. This part is divided into four chapters. The first chapter (chapter 3) provides a detailed summary of the contribution of this research work. The second chapter (chapter 4) deals with the research in the context of the MPEG-21 standard, mainly related to integration activities amongst different parts of the standard, such as DID, REL, RDD, DIP, and ER. The third chapter (chapter 5) specifies the MIPAMS generic DRM architecture and, after providing mappings and comparisons with other initiatives, it focuses on three different projects that have tackled its implementation: VISNET II [19], GILDDA [20] and AXMEDIS [21]. The fourth chapter (chapter 6) deals with Linked-Work, a specific architecture devised for the registration of original and derived digital content. Although Linked-Work architecture follows in general terms the structure of MIPAMS, it has been included in a different chapter because it provides some advanced functionality that is not usually present in a DRM architecture.

The conclusion part of the document provides a summary of the most relevant contributions in the research work and details the publications were it has been presented. Moreover, it gives an overview of the future research lines that are foreseen for the next years.

Part II. State of the Art

2 DRM systems

2.1 Introduction

Content creators, distributors and consumers have currently plenty of formats and protocols for coding, distributing or accessing multimedia content and services from almost anywhere at anytime. However, the lack of a global end-to-end solution that allows the different actors to interact with multimedia content in an interoperable way is slowing down the deployment of advanced multimedia creation and distribution applications, although the technology for doing so is already available.

From the end users perspective, access devices have different terminal and network capabilities and they are used in different locations and environments. Users, however, do not have currently the appropriate tools to deal efficiently with the complex multimedia usage environment.

Digital media is every time more used not only for personal use but also for sharing among a wider range of users. Everyone can, thus, act as a content provider, having to deal with some matters such as the management of content, the adaptation of content based on consumer or device capabilities, the protection of rights and the protection from unauthorised access or modification, among others.

Such developments provide new business models for trading digital content where it is becoming increasingly difficult to identify the different intellectual property rights that are associated with different content types (e.g. audio, video, text, images) in multimedia content. New solutions are required to manage the access and delivery process of these different content types in an integrated and harmonised way, entirely transparent to the user of multimedia services.

In next sections, we will analyse different standards and initiatives that define different ways of tackling the distribution of protected content using a Digital Rights Management (DRM) approach. We will mainly focus on the aspects that are closely related to this research work.

2.2 MPEG-21

In June 2000, MPEG (ISO/IEC JTC1 SC29 WG11) started to work on the definition of enabling normative technology for the multimedia applications of the 21st century: MPEG-21 “Multimedia Framework” [2]. MPEG-21’s approach is to define a framework to support transactions that are interoperable and highly automated, specifically taking into account Intellectual Property Management and Protection requirements and targeting multimedia access and delivery using heterogeneous networks and terminals.

MPEG-21 aims at defining a normative but open framework for multimedia creation and sharing that can be used by all the players in the creation, delivery and consumption chain. This open framework will provide content creators and service providers with equal opportunities in the MPEG-21 enabled open market. This will also be to the benefit of the content consumer providing them access to a large variety of content in an interoperable manner.

MPEG-21 identifies and defines the normative technologies needed to support the multimedia delivery chain as described above as well as the relationships between and the operations supported by them. Within the parts of MPEG-21, these elements are elaborated by defining the syntax and semantics of their characteristics, such as interfaces to the elements.

The MPEG-21 multimedia framework has two essential concepts: Digital Items and Users.

A Digital Item is a structured digital object with a standard representation, identification and metadata within the MPEG-21 framework. This entity is the fundamental unit of distribution and transaction within this framework. In practice, a Digital Item is a combination of resources and metadata in a structured way. The resources are the content. The metadata comprises information about the Digital Item as a whole or about the individual Resources included in the Digital Item. Finally, the structure relates to the relationships among the parts of the Digital Item, both resources and metadata. An example of a Digital Item may be a music album including the music together with photos, videos, lyrics, interviews with the singers, etc.

In MPEG-21 a User is any entity that interacts in the MPEG-21 environment or makes use of Digital Items, independently of the role it plays, as for example “content provider” or “consumer”. However, the rights a User has may depend on their interaction with other Users within MPEG-21. Some such interactions are: creating content, providing content, modifying content, archiving content, aggregating content, delivering content and consuming content.

The MPEG-21 standard is divided currently into seventeen parts, which deal with different aspects of multimedia information management:

Part 1: Vision, Technologies and Strategy [22]. The purpose of this part of the standard is to define a vision for a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices to meet the needs of all users. This part has as objective to achieve the integration of standards to facilitate harmonisation of technologies for the creation, management, distribution and consumption of digital items. Moreover, it shall define a strategy for achieving a multimedia framework based on well-defined functional requirements through collaboration with other bodies.

Part 2: Digital Item Declaration (DID) [3]. Part 2 of MPEG-21 describes a set of abstract terms and concepts to form a useful model for defining Digital Items. Within this model, a Digital Item is the digital representation of an Asset, and as such, it is the thing that is acted upon (managed, described, exchanged, collected, etc.) within the model. Moreover, it also includes a normative description of the syntax and semantics of each of the Digital Item Declaration elements, as represented in XML, and a normative XML schema comprising the entire grammar of the Digital Item Declaration representation in XML. Part 2 of MPEG-21 defines a model for defining Digital Items, and the Digital Item Declaration Language (DIDL) grammar used for its representation in XML documents. In other words, it defines the structure of the Digital Item, which comprises the Digital Item Declaration (DID) and the Resources.

Part 3: Digital Item Identification (DII) [23]. This part of the standard provides a means to include already existing identification systems into a Digital Item Declaration in order to identify Digital Items and parts thereof (such as resources).

Part 4: Intellectual Property Management and Protection (IPMP) [24]. This part deals with the standardisation of a general solution for the management and protection of Intellectual Property. Digital Items can be protected in order for preventing the unauthorised access to the content. The solution lies in the use of protection techniques over the digital content, which makes it possible to deploy a business model that ensures the accomplishment of the license terms in a controlled way. These kinds of objects are called IPMP DIDL documents that consist of the protected object (or part of the DIDL document) and the IPMP information expressions. IPMP expressions contain protection information, such as the IPMP tools that protect the content, initialisation settings, keys, etc.; and governance information, such as licenses that govern the content or references to these licenses or license services.

Part 5: Rights Expression Language (REL) [4]. A Rights Expression Language is seen as a machine-readable language that can declare rights and permissions using the terms as defined in the Rights Data Dictionary. RELs are used to create Digital licenses, which can be seen as digital documents that establish a contractual relationship between two parties: the license issuer and the granted party. Digital licenses are intended to provide flexible, interoperable mechanisms to support transparent use of digital resources while taking into consideration the rights, fees and conditions of use specified for digital content. RELs, through digital licenses, are also intended to support specification of access and use controls for digital content in cases where financial exchange is not part of the terms of use, and to support exchange of sensitive or private digital content.

Part 6: Rights Data Dictionary (RDD) [5]. The RDD comprises a set of clear, consistent, structured, integrated and uniquely identified Terms to support the MPEG-21 Rights Expression Language. The Dictionary is a prescriptive Dictionary, in the sense that it defines a single meaning for a Term represented by a particular RDD name (or Headword), but it is also inclusive in that it recognises the prescription of other Headwords and definitions by other Authorities and incorporates them through mappings. It is based on the use of verbs that are contextualised so that a dictionary created using the model with it can be as extensible and granular as required.

Part 7: Digital Item Adaptation (DIA) [25]. This part of MPEG-21 specifies the syntax and semantics of tools that may be used to during the adaptation of Digital Items, i.e., the Digital Item Declaration and resources referenced by the declaration. These tools could be used to produce the adapted Digital Items that satisfy transmission, storage and consumption constraints, as well as Quality of Service management by the various Users. Although Digital Items are subject to a resource adaptation engine, as well as a descriptor adaptation engine, which together produce the adapted Digital Items, it is important to emphasise that the adaptation engines themselves are non-normative tools of this part of MPEG-21.

Part 8: Reference Software [26]. This part of MPEG-21 describes reference software implementing the normative clauses of the other parts of the standard. The information provided is applicable for determining the reference software modules available for parts of MPEG-21, understanding the functionality of the available reference software modules, and utilising the available reference software modules. In addition to the reference software, available utility software is also described. This utility software can assist in understanding how to use the reference software.

Part 9: File Format [27]. ISO/IEC 21000-9:2005, the MPEG-21 File Format, defines an open framework for multimedia delivery and consumption, with both the content creator and content consumer as focal points. The vision for MPEG-21 is to define a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities. ISO/IEC 21000-9:2005 is designed to contain a base MPEG-21 XML document with some or all of its ancillary resources, potentially in a single package. It forms part of a family of specifications which are box-structured, and is built using tools from the ISO Base Media File Format ISO/IEC 14496-12 (technically identical to 15444-12), specifically those that provide the unified structural approach to both static meta-data (untimed meta-data) and MPEG-21 integrated document handling.

Part 10: Digital Item Processing (DIP) [6]. This part of the standard specifies the syntax and semantics of tools that may be used to process Digital Items: 1) Digital Item Methods: Tools enabling users to include sequences of instructions for adding predefined functionality to a Digital Item. Digital Item Methods are authored with the Digital Item Method Language, an extension of ECMAScript language (ISO/IEC 16262), which includes a normative set of Digital Item Base Operations (DIBOs). The predefined functionality specified by a Digital Item Method provides a suggested interaction between a User and the Digital Item. 2) Digital Item eXtension Operations (DIXOs): Provide for extended functionality not included by the normative set of Digital Item Base Operations to be implemented efficiently in a higher level programming language. 3) Linkage with ISO/IEC 21000-2: Tools for integrating Digital Item Methods and Digital Item eXtension Operations with Digital Item Declarations (as specified by ISO/IEC 21000-2).

Part 11: Evaluation Tools for Persistent Association Technologies [28]. MPEG-21 provides a framework within which many elements of multimedia are brought together. In particular, coded representations of content are juxtaposed with metadata descriptors and IPMP protection that apply to the content. This leads to a requirement for tools that can create and maintain (e.g. detect or extract) an association between content, metadata and IPMP elements within MPEG-21. ISO/IEC TR 21000-11:2004 describes methodologies for the evaluation of two classes of technologies that can create and maintain such associations: "watermarks" and "fingerprints", when applied to audio content.

Part 12: Test Bed for MPEG-21 Resource Delivery [29]. This part specifies a test bed that is designed to assist in performance assessment of MPEG-21, Scalable Video Codec (SVC) for streaming applications, and for the evaluation of resource delivery technologies over unreliable packet-switched networks. The streaming protocols used in the test bed are based on RTSP and RTP. The Network Adaptation QoS mechanism of MPEG-21 Digital Item Adaptation (DIA) is used for bandwidth-scalable streaming. A subset of MPEG-4 IPMP is also included in the test bed so that encrypted streaming and layered access functionality of a DRM system can be tested for different SVC designs.

Part 14: Conformance Testing [30]. The purpose of this part is to specify conformance points and conformance tests for different parts of ISO/IEC 21000. Based on the various conformance points, it is identified which requirements defined in ISO/IEC 21000 apply to those conformance points. The tests are developed to ascertain whether a particular artifact (such as a piece of software or hardware or a document) meets all the requirements for a specific conformance point or not.

Part 15: Event Reporting (ER) [31]. Event Reporting specifies a mechanism to monitor events associated with the manipulation and usage of Digital Items, as defined in ISO/IEC 21000-2, and Peers. Monitoring the usage of audio-visual digital material and gaining insight into the state or capacity of a Peer is an important functionality for many content creation, delivery, adaptation and consumption applications. It specifies a dynamic mechanism which allows Users to create an Event Report Request within a Digital Item which can then be processed by a Peer. Such an Event Report Request specifies the conditions when an Event Report will be first generated and then sent to a (set of) recipient Peer(s).

Part 16: Binary Format [32]. This part of the standard specifies the binary format to efficiently serialise XML-based descriptions as specified within other ISO/IEC 21000 parts. The MPEG-21 binary format enables the efficient interchange or storage of ISO/IEC 21000 descriptions.

Part 17: Fragment Identification of MPEG Resources [33]. This part of the standard specifies a normative syntax for Fragment Identifiers to be used in URIs (Uniform Resource Identifiers) for addressing parts of any resource whose Internet Media Type is one of: audio/mpeg; video/mpeg; video/mp4; audio/mp4; application/mp4.

Part 18: Digital Item Streaming [34]. This part of the standard specifies tools for Digital Item Streaming. The first tool is the Bitstream Binding Language, which describes how Digital Items (comprising the Digital Item Declaration, metadata and resources) can be mapped to delivery channels such as MPEG-2 Transport Streams or the Real-time Transport Protocol.

In subsequent sections, we will present a deeper analysis of the MPEG-21 parts that are more closely related to our research work. Therefore, we will have a more detailed look at the essential concepts needed to understand the contribution of this research work, which will be presented in the second part of this document.

2.2.1 Digital Item Declaration (DID)

The Digital Item Declaration (DID) Part of MPEG-21 describes the technology for creating Digital Items. A Digital Item is defined in [3] as a structured digital object, including a standard representation, identification and metadata. It is the fundamental unit of distribution and transaction inside MPEG 21. The DID part is divided into three normative clauses: Model, Representation and Schema.

The DID Model clause describes a set of abstract terms that build an abstract model into which the information of the Digital Items can be mapped. This model is used as the basis for the definition of the DID grammar.

Some of the terms are defined in the model clause, as the following:

- Container: A *container* is a structure that allows *items* and/or *containers* to be grouped. These groupings of *items* and/or *containers* can be used to form logical packages (for transport or exchange) or logical shelves (for organisation). *Descriptors* allow for the "labelling" of *containers* with information that is appropriate for the purpose of the grouping (e.g. delivery instructions for a package, or category information for a shelf).

- Item: An *item* is a grouping of sub-*items* and/or *components* that are bound to relevant *descriptors*. *Descriptors* contain information about the *item*, as a representation of a work. *Items* may contain *choices*, which allow them to be customised or configured. *Items* may be conditional (on *predicates* asserted by *selections* defined in the *choices*). An *item* that contains no sub-*items* can be considered an entity -- a logically indivisible work. An *item* that does contain sub-*items* can be considered a compilation -- a work composed of potentially independent sub-parts. *Items* may also contain *annotations* to their sub-parts.

In this way, the prose descriptions of the different terms define their semantic meaning. An EBNF representation is also provided for each term, so that the relationship between the different terms in the model is formally described.

Other terms defined in the model are summarised below:

- Component: it is the binding of a resource to a set of descriptors.
- Anchor: it binds descriptors to a fragment, which corresponds to a specific location or part of a resource.
- Descriptor: it associates information such as a textual statement or an image with the enclosing element.
- Condition: it describes the enclosing element as being optional, and links it to the selection(s) that affect its inclusion.
- Choice: it describes a set of related selections that can affect the configuration of an item.
- Selection: it describes a specific decision that will affect one or more conditions somewhere within an item. If the selection is chosen, its predicate becomes true; if it is not chosen, its predicate becomes false; if it is left unresolved, it is undecided.
- Annotation: it describes a set of information about another identified element of the model without altering or adding to that element.
- Assertion: it defines a full or partially configured state of a choice by asserting true, false or undecided values for some number of predicates associated with the selections for that choice.
- Resource: it is an individually identifiable asset such as a video or audio clip, an image, a textual asset or even a physical object.
- Fragment: it designates a specific point or range within a resource.
- Statement: it is a literal textual value that contains information, but not an asset.
- Predicate: it is an identifiable declaration that can be true, false or undecided.

Figure 1 summarises the most important elements within the model, their relationship and the hierarchical structure of the Digital Item Declaration Model. The represented digital item consists of a container, which inside groups some items together with their descriptors and components.

Based on the declaration model, the DID Part describes the syntax and semantics of the XML representation for declaring Digital Items. Digital Item Declaration Language (DIDL) documents are XML 1.0 documents. In addition, DIDL syntax is based on an abstract structure defined in the Digital Item Declaration Model. The following abstract elements defined in the Model are each represented in DIDL by a DIDL element that shares the same name: *container, item, component, anchor, fragment, descriptor, choice, selection, condition, annotation, assertion, resource* and *statement.*

For example, the abstract *descriptor* entity in the Model is represented in DIDL by the DESCRIPTOR element.

In addition, DIDL defines two element types that do not correspond to any of the Model entities: REFERENCE, and DECLARATIONS. The REFERENCE element is used to link the contents of an element inside another element. References can be made to elements within a document, or to elements in a different document. The DECLARATIONS element is used to define a set of DIDL elements in a document without actually instantiating them. A declared element (i.e. a child element of a DECLARATIONS element) is not considered to be instantiated unless it is referenced (by a REFERENCE element).

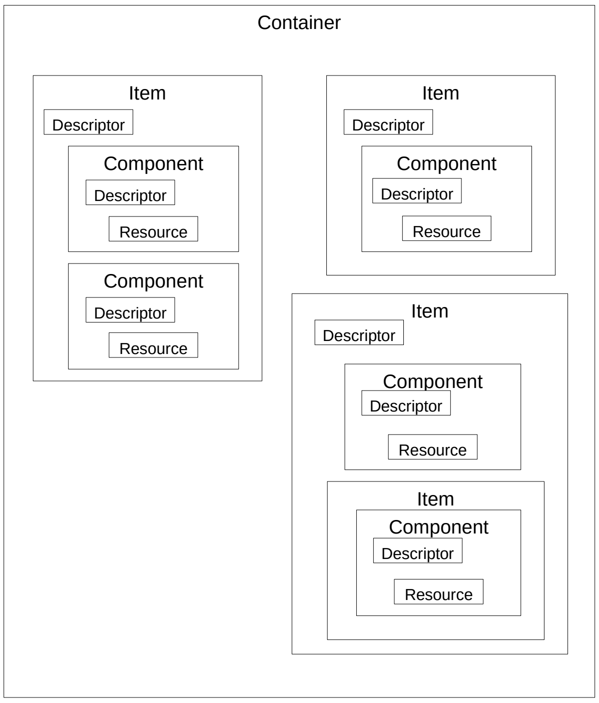


Figure 1. Integral elements in the Digital Item Declaration Model. Source: [3]

An example of how MPEG-21 defines the representation of the ITEM element is shown below:

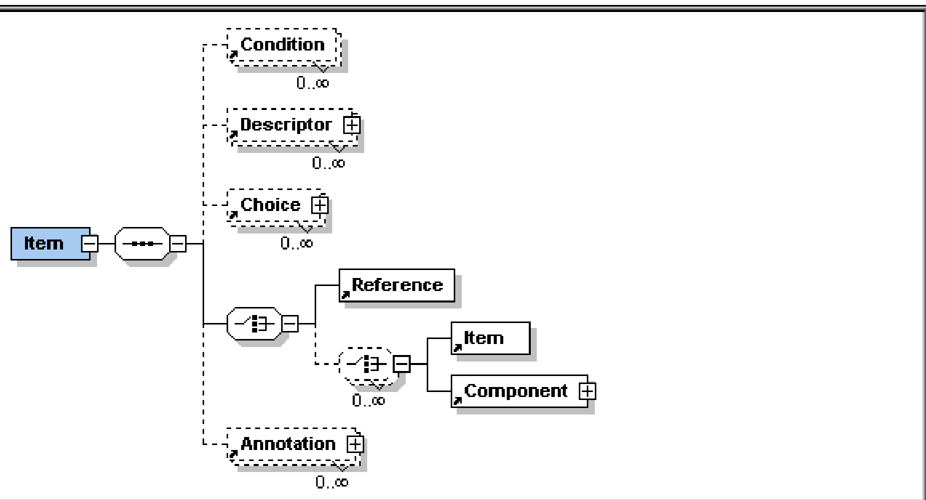
<Item>

An ITEM element represents an *Item.* As such, it is a grouping of possible sub-ITEMS and/or COMPONENTS, bound to a set of relevant DESCRIPTORS containing descriptive information about the *item*. In addition, an ITEM can be made conditional via a set of CONDITION child elements, made configurable via a set of CHOICE elements, and annotated via a set of ANNOTATION elements.

Items are intended to be the lowest level of granularity transacted by Users within the MPEG-21 framework.

Validation Rule:

An ITEM element cannot be conditional on any of its descendant SELECTION elements. In other words, an ITEM cannot contain a CONDITION element specifying a SELECT_ID value that identifies any descendant SELECTION element within the ITEM.

Diagram	 Item Condition 0..∞ Descriptor 0..∞ Choice 0..∞ Reference Item Component 0..∞ Annotation 0..∞		
Children	<Condition> <Descriptor> <Choice> <Reference> <Item> <Component> <Annotation>		
Used by	<Declarations> <Item> <Container> <DIDL>		
Attributes	**Name**	**Type**	**Description**
	id	ID	A unique ID value.

Source

```
<xsd:element name="Item">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element ref="Condition" minOccurs="0"
                                maxOccurs="unbounded"/>
                        <xsd:element ref="Descriptor" minOccurs="0"
                                maxOccurs="unbounded"/>
                        <xsd:element ref="Choice" minOccurs="0"
                                maxOccurs="unbounded"/>
                        <xsd:choice>
                                <xsd:element ref="Reference"/>
                                <xsd:choice minOccurs="0"
                                                maxOccurs="unbounded">
                                        <xsd:element ref="Item"/>
                                        <xsd:element ref="Component"/>
                                </xsd:choice>
                        </xsd:choice>
                        <xsd:element ref="Annotation" minOccurs="0"
                                maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:attributeGroup ref="ID ATTRS"/>
        </xsd:complexType>
</xsd:element>
```

Figure 2. Representation of the ITEM element in MPEG-21. Source: [3]

As we have seen, the representation of an element consists of a textual description of the element plus some validation rules, the XML schema that represents the element, the attributes it has and a graphic diagram that summarises its structure.

In order to check the validity of a DIDL document, it is not sufficient to validate it against the DIDL Schema. It must also be subjected to additional validation rules, which are given in the descriptions of the elements to which they pertain.

An example of a very simple DIDL document is shown below. The Digital Item consists of a locally referenced video plus a reference to an external web page.

```
<?xml version="1.0" encoding="UTF-8"?>
<didl:DIDL xmlns:didl="urn:mpeg:mpeg21:2002:01-DIDL-NS" xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance xsi:schemaLocation="urn:mpeg:mpeg21:2002:01-
DIDL-NS ../Schemas/DIDL-AMD1.xsd urn:mpeg:mpeg21:2002:01-DII-NS ../Schemas/dii.xsd">
    <didl:Item>
```

```
        <didl:Component>
            <didl:Descriptor>
                <didl:Statement mimeType="text/plain">
                    Holidays in Alaska
                </didl:Statement>
            </didl:Descriptor>
            <didl:Resource mimeType="video/mpeg" ref="video.mpg"/>
        </didl:Component>
        <didl:Component>
            <didl:Descriptor>
                <didl:Statement mimeType="text/xml">
                    <dii:Identifier>urn:grid:a1-abcde-9630741852-f</dii:Identifier>
                </didl:Statement>
            </didl:Descriptor>
            <didl:Resource mimeType="text/html" ref="http://myholidays.foo/alaska_holidays.html"/>
        </didl:Component>
    </didl:Item>
</didl:DIDL>
```

Figure 3. Example of DIDL document.

The DIDL Representation defines another specific element: the STATEMENT element.

The possibility of inserting data in any kind of data format, specially well-formed XML, inside a Statement provides a wide field for inserting information for the protection and identification of multimedia information. For example, if we want to associate rights expressions to a particular resource within a Digital Item, the REL License can be placed in the Statement of the Descriptor element related to the resource.

However, the IPMP part of MPEG-21 has recently defined a specific set of elements where to place the licenses. These specific elements are used to associate the license together with the protection mechanisms to govern a Digital Item or the parts thereof

2.2.2 MPEG-21 REL and RDD

Parts 5 and 6 of MPEG-21 describe its Rights Expression Language (REL) [4] and Rights Data Dictionary (RDD) [5], respectively. These are the fundamental parts of the DRM based on MPEG-21 as they define the licenses, which include the rights that users may have over Digital Items (DI). Digital Items (Part 2 of MPEG-21 standard) are the fundamental unit of distribution and transaction inside MPEG-21.

Right Expression Languages (RELs) are languages devised to express conditions of use of digital content. They have been proposed to describe licenses governing digital content. Part 5 of the MPEG-21 standard specifies the syntax and semantics of a Rights Expression Language. MPEG chose XrML [35] as the basis for the development of the MPEG-21 REL. It makes use of the RDD, which comprises a set of clear, consistent, and structured terms. The RDD defines the meaning for the terms defined in the REL.

MPEG-21 REL specifies the syntax and semantics of the language for issuing rights for users to act on DIs. The most important concept in REL is the license (see Figure 1) that conceptually is a container of grants, each one of which conveys to a principal the sanction to exercise a right against a resource. A license if formed by the elements *title*, *inventory*, *grant* or *grantGroup* and *otherInfo*. The *title* element provides the description about the license. The *inventory* element is used for defining variables within a license. The *grant* and *grantGroup* elements of a license convey some rights to an entity, which are subject to certain conditions. A *grant* is formed by four elements:

- a *principal*, which represents the entity involved in the granting or exercising of rights.

- a *right*, which specifies an action or activity that a principal may perform.
- a *resource*, which represents the object against which the principal has the rights.
- a *condition*, which represents grammatical terms, conditions and obligations that a principal must satisfy.

The *issuer* element contains two pieces of information, the identification of the issuer and a set of specific details about the issuance of the license. The *otherInfo* element can be used to place additional non-normative information.

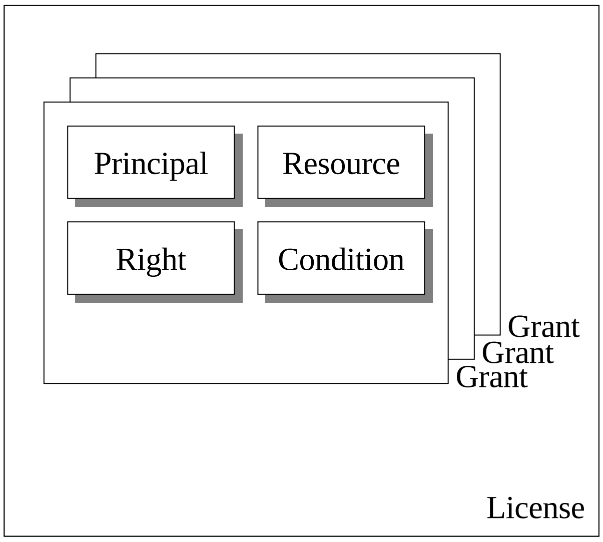


Figure 4. REL License Structure.

The principals, rights, resources and conditions of the REL are organised in three main groups. The first one, the Core specifies structural elements and types and how are they related. The standard extension and the multimedia extension specify a standard means for expressing multimedia principals, rights, resources and conditions.

The MPEG-21 REL can be extended to support new business models defining extensions. The extension mechanism specified in MPEG-21 REL allows the addition of new elements to address the requirements of a new application domain.

Currently, the MPEG-21 REL standard specification has five extensions: multimedia, standard, multimedia extension one, multimedia extension two and multimedia extension three. The standard extension defines terms to extend the usability of the core schema; essentially, it defines conditions that restrict the use of the content, for example in the interval of time, number of times that it can be used, the fees that must be paid, the territory, etc. The multimedia extension expands the core schema by specifying terms that relate to digital works. It describes rights, conditions and metadata for digital works and includes new rights such as modify, enlarge, reduce, move, adapt, play, print, execute, etc. resources as DIs. The other three extensions have been defined to support the MPEG-21 REL profiles.

On the other hand, three profiles have been specified and included in this part of the standard as amendments to the MPEG-21 REL standard. The first one, called Mobile And optical Media (MAM) profile [36], addresses the needs of the mobile and optical media domains. Moreover, it facilitates the interoperability with OMA DRM REL v2 [7]. In order to support the requirements of this profile, the “multimedia extension one” profile was defined with new rights and conditions for the pre-recorded optical media and mobile domain. This profile consists of a subset of the elements defined in the core and in the multimedia and standard extensions and the rights and conditions defined in the multimedia extension one. The second one, the Dissemination and Capture (DAC) profile [37] was designed to be able to represent the concept of the OMA DRM v2.0 Extensions for Broadcast Support and to facilitate the interoperability with the TV-Anytime Rights Management and Protection information [38]. In order to support the requirements of this profile, the “multimedia extension two” profile was defined with new rights and conditions for the broadcast domain. The third one, the Open Release Content (ORC) profile [39] is under development. It has been defined to support the different types of Creative Commons [40] licenses. In order to support the requirements of this profile, the “multimedia extension three” profile is under development with new rights and conditions for the open release domain.

2.2.3 MPEG-21 IPMP

The Intellectual Property Management and Protection (IPMP) Components, part 4 [24] of the MPEG-21 standard, deals with the standardisation of a general solution for the management and protection of Intellectual Property.

This part of MPEG-21 defines a language to provide protection and governance (i.e. control of content usage rights and conditions by means of a digital license) to any part of a Digital Item (DI), from a complete DI to a specific asset. The IPMP DIDL protects a part of the hierarchy of a DI, and provides mechanisms to associate appropriate identification and protection information to the protected part. Each of the IPMP DIDL elements contains the following elements: Identifier, Info, ContentInfo and Contents. The Identifer element contains a unique identifier for the protected element. The Info element contains information about the protection tools and the rights expressions that govern the protected element. The ContentInfo element contains information about the protected element. Finally, the Contents element acts as a placeholder for the protected contents.

The IPMP Components element also defines the information regarding the protection of a DI. This information falls into two categories: information about protection and governance related to the whole DI and information about the specific protection applied to a certain part of a protected DI. The general protection information includes the collection of licenses and a list of the protection tools that have been used, which can be later referred from specific protected elements. On the other hand, the specific information includes the specific tools and protection keys that have been applied, the licenses for that content, etc.

2.2.4 MPEG-21 Event Reporting

Part 15 of MPEG-21 Event Reporting (ER) [31] is required within the MPEG-21 Multimedia Framework to provide a standardised means for sharing information about Events amongst Peers and Users. An Event, which can be defined as the occurrence of a reportable activity, is related to Digital Items and/or Peers that interact with them. In the MPEG-21 context, the reporting messages that include information about different aspects of media usage are called Event Reports.

Event Reporting could be useful when monitoring the usage of copyrighted material. The provider offering Digital Items for download would specify in an Event Report Request that, whenever a Resource within a Digital Item is rendered (e.g. played), they would receive an Event Report enabling them to manage their royalties. Upon rendering, the Peer would generate an ISO/IEC 21000 Event Report which would be delivered to the rights holder specified in an Event Report Request, containing information about the Digital Item, the Resource, and the conditions under which it would have been rendered.

Fundamentally, Event Reporting facilitates interoperability between consumers and creators, thereby enabling multimedia usage information to be both requested and represented in a normalised way. Examples where Event Reports may be requested include usage reports, copyright reports, financial reports and technical reports.

The basic model of Event Reporting indicates that the Events that need to be reported may be specified by interested parties through the use of an Event Report Request (ERR). An ERR is used to define the conditions under which an Event is deemed to have occurred. Events defined by ERRs trigger the creation of an associated Event Report (ER), which contains information describing the Event, as specified in the associated ERR.

The ER purpose is to indicate which Peer created it, define the data items that are to be included in such Event Reports, provide a reference to the originating ERR and provide status information regarding its completion and creation, along with a freeform description.

Although the MPEG-21 standard specifies the ERR format, it is worth noting that it is not normative that an ER is only created as the result of the processing of an ERR. This means that applications may create Event Reports which are normative on their own initiative.

Event Reports contain three main Elements. They are used to provide description of the Event Report, to contain the Event Report's payload and to optionally contain an Embedded Event Report Request.

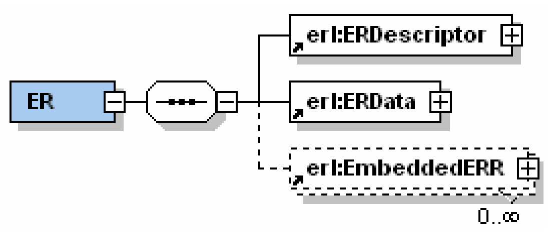


Figure 5. MPEG-21 Event Report.

These three Elements, which are depicted in Figure 5 are, in more detail, the following:

- ERDescriptor: It contains a free-form string field, used to convey information on whether the Peer was able to compliantly generate the Event Report and finally information regarding the creation of the Event Report.
- ERData: It contains the "payload" data of the Event Report, which describes the performed action or operation.
- EmbeddedERR: It contains an Event Report Request that that has been included within and is associated with the Event Report.

Next, we are going to briefly describe the fields that are standardised for ERDescriptor and ERData fields, whereas in subsequent chapters we will see which are the implications of its usage in a multimedia content distribution and consumption environment.

Event Report ERDescriptor fields

Figure 6 depicts Event Report Descriptor fields, which are described in more detail in Table 1.

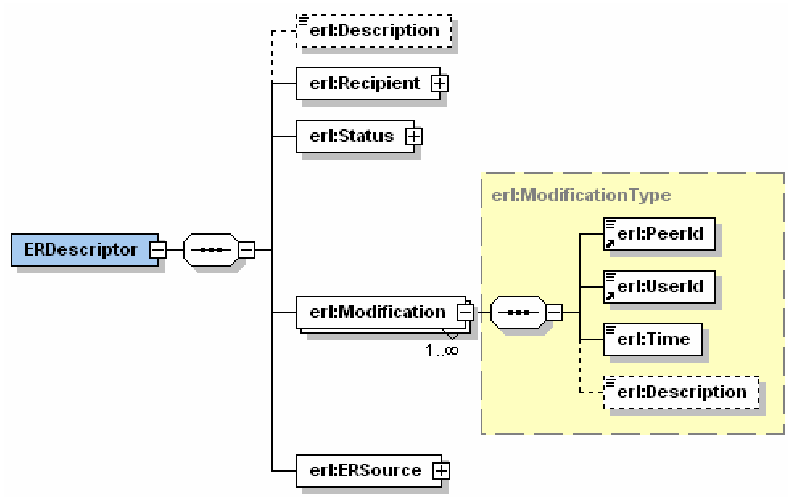


Figure 6. MPEG-21 Event Report Descriptor.

Table 1. MPEG-21 Event Report Descriptor fields description.

Field		**Description**
Description		Free form field
Status		Denotes its completion status
Modification		Reports who has modified it
	PeerId	Identifier of the peer that created or modified the ER
	UserId	Identifier of the user that created or modified the ER
	Time	When the ER was created or modified
Recipient		Intended recipient of the ER
ERSource		Original source that created the ER
	ERR	Inline ERR source of the ER
	ERRReference	Referenced ERR source of the ER
	OtherSource	Reference to application that is the source of the ER

Event Report ERData fields

Figure 7 depicts the Event Report Data fields, which are described in more detail in Table 2.

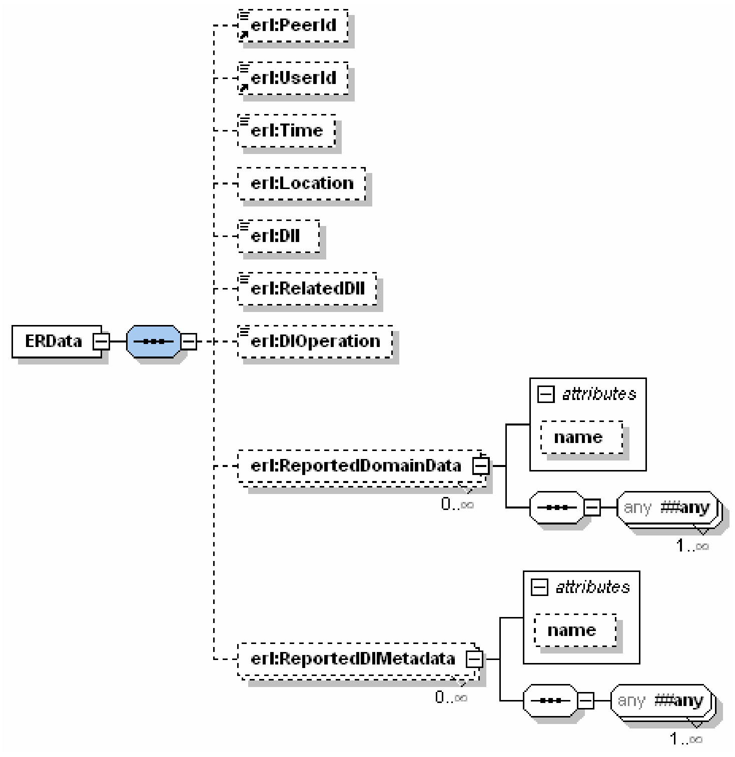


Figure 7. MPEG-21 Event Report Data.

Table 2. MPEG-21 Event Report Data fields description.

Field		**Description**
PeerId		Identifier of the peer that created the ER
UserId		Identifier of the User that was using the Peer that created the ER
Time		Time as returned by the Peer
Location		Location information of Peer
DII		Referenced DI identifier
RelatedDII		Related DI identifier
DIOperation		Operation performed by the User
ReportedDomainData		Domain-specific data item elements
	semantics	"meaning" of the reported ReportedDomainData
ReportedDIMetadata		Meta-data items associated with the DI
	name	Specific name of the Meta-data field being reported.

2.2.5 Digital Item Processing (DIP)

Digital Item Processing (DIP) [6] specifies the syntax and semantics of tools that may be used to process Digital Items. The objective of DIP is to provide a normative set of tools for specifying the processing of a Digital Item in a predefined manner. In this way, it is be possible to extend the Digital Item Declaration Language [3] in order to add user specific functionality inside the Digital Item, but maintaining the interoperability at the processing level.

A key component of Digital Item Processing is the Digital Item Method (DIM), that is, the mechanism that enables Digital Item Users to include sequences of instructions for adding predefined functionality to a Digital Item. The method definition may be referenced from or embedded in the Digital Item Declaration (DID). For example, a Digital Item representing a music album may contain a DIM to add a new music track to the album. Such a DIM can be used to ensure that the new music track is added to the Digital Item while maintaining a predefined format (i.e. elements added in the correct place in the DID structure, correct Descriptors are included, etc.).

The way in which users interact with a Digital Item using DIP depends on the implementation and could be as follows:

- On receipt of a DID, a list of Digital Item Methods that can be applied to the Digital Item can be made available to the User. The User can choose a Digital Item Method that is then executed by the Digital Item Processing engine.
- On receipt of a Digital Item Declaration, a list of Objects is presented based on the presence of Identifiers of the DII XML Namespace. The User chooses one or more of these Object(s). A list of Digital Item Methods that takes as arguments the (set of) Object(s) is then presented to the User. The User selects a Digital Item Method that is then executed by the Digital Item Processing engine.

A DIM is expressed using the Digital Item Method Language (DIML), which includes a binding for Digital Item Base Operations (DIBO). The Digital Item Method Language provides the basic syntax, control flow constructs, etc for authoring a Digital Item Method.

The Digital Item Base Operations (DIBOs) are the functional building blocks utilised by a Digital Item Method. They can be considered somewhat analogous to the standard library of functions of a programming language. A DIBO is described by a normatively defined interface and semantics, while the DIBO implementation will depend on the peculiarities of the terminal in which it is to be executed.

Initially in the DIP SoCD [41], the standardised DIBOs fell into one of the following categories:

- Operations that manipulate the DID at the DIDL level. For example, DIBOs that add child nodes, remove child nodes, modify element attributes, etc.
- Operations that manipulate the state of a DI (note that at the DID level the DI state is defined by the state of the DIDL Choice elements). For example, DIBOs that set the state of a DIDL Selection element within a Choice element, resolve DIDL Condition elements predicated on the state of Choice elements, etc.
- Operations that are representations of or associated with an RDD verb. For example, DIBOs that apply the Play verb to a resource, the Store verb to a DID, etc.

On later specifications [6], DIP has evolved and currently DIML includes ECMAScript bindings to support the whole DOM Level 3 Core API. Moreover, currently DIBOs are grouped into different categories, depending on the relation of their functionality with the different parts of MPEG-21. In this way, we have for example DID related DIBOs, as ConfigureChoice and SetSelection, DIP related DIBOs, as Play and RunDIM, etc.

Table 3 presents the Interface defined for a DIBO that manipulates the DID at the DIDL level, whereas Table 4 does so for a DIBO associated to a RDD verb. It is worth saying that apart from the DIBO interface, DIP also defines its standard semantics.

Table 3. GetDIDLNode DIBO Interface.

Syntax:	`GetDIDLNode(docLocation, rootNode)`
Description:	Retrieves an existing DIDL element from the source DID containing the `rootNode`.
Parameters:	`docLocation` A `String` containing an XPath expression specifying the location of the element to retrieve.
	`rootNode` If the XPath is relative, this argument is the `MpegDIDNode` object to be used as the root element of the XPath expression.
Return value:	An array of `MpegDIDNode` objects retrieved by the DIBO. The array may be empty if no `MpegDIDNode` objects are retrieved by the specified `docLocation`.
Exceptions:	None.

Table 4. PlayResource DIBO Interface.

Syntax:	`PlayResource(resourceNode, changes, async)`
Description:	Plays the specified media resource and optionally wait for completion of the media resource before returning control to the calling DIM. Optional changes to be applied when playing the resource may be provided.
Parameters:	`resourceNode` The `MpegDIDNode` object that represents the DIDL RESOURCE element describing the media resource.
	`changes` An `MpegDIPResourceChangeObject` containing changes to be applied to the resource when being played. A copy of `changes` shall be made and utilised. This parameter may be null in which case no changes are applied.
	`async` A `Boolean` indicating if the resource should be played asynchronously or not. If `true` then the resource is played asynchronously and the DIBO should return control immediately to the calling DIM after playing of the resource is initiated. If `false` then the resource is played synchronously and control is not returned to the calling DIM until the resource media end time has been reached.
Return value:	Returns an `MpegDIPResourceStatus` object to identify the playing resource. This is included so that the resource may be stopped or paused at a later time.
Exceptions:	None.

Figure 8 shows the code of a PlayContent DIM. This DIM takes as an input an Item (or node) from which it extracts the first resource it finds through the GetDIDLNode DIBO. Next, it extracts the embedded licenses in the DIDL document, if any, through the GetLicense DIBO, and finally if there is a license, it tries to play the resource through the PlayResource DIBO, which is responsible of performing the corresponding authorisation based on the license terms.

```
Function PlayContent(Item)
{
        var resource=GetDIDLNode("didl:Component/didl:Resource", Item);
        if(resource!=null){
                var license=GetLicense(resource[0]);
                if(license!=null){
                        PlayResource(resource[0],license,false);
                }
                else Alert("No license embedded in the DIDL document", -1);
        }
}
```

Figure 8. PlayContent DIM.

Digital Item eXtension Operations specify a normative mechanism for enabling functionality that extends beyond the basic functionality of the normative set of DIBOs in an efficient way. As well as DIMs, DIXOs may be referenced from or embedded in the DID. DIXOs will be used when Users need to extend base operations when such operations are: not normatively defined by ISO, unique to the application space, useful only for this DI or too sub-optimal to be done using DIML.

An example of a DIXO is the `TypeText` DIXO. This DIXO could implement the functionality of displaying text *character by character* (i.e., one letter at a time). Other possible uses are: fading in and/or fading out; font effects; blinking; colour; etc. In other words, the `TypeText` DIXO can allow for text display with extended functionality, as opposed to the base (text) display mechanisms that are provided by the normatively specified DIBOs.

Figure 9 summarises the relationship between DIMs, DIBOs and DIXOs.

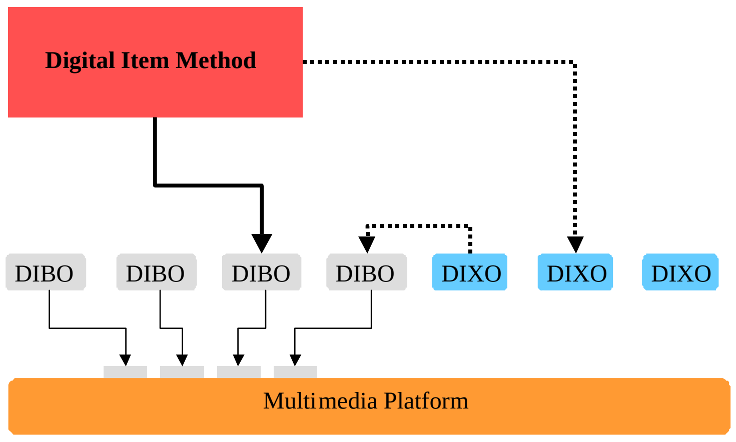


Figure 9. Relationship between DIMs, DIBOs and DIXOs. Source: [41]

DIP provides a XML Schema to structure the information provided by DIMS, etc.

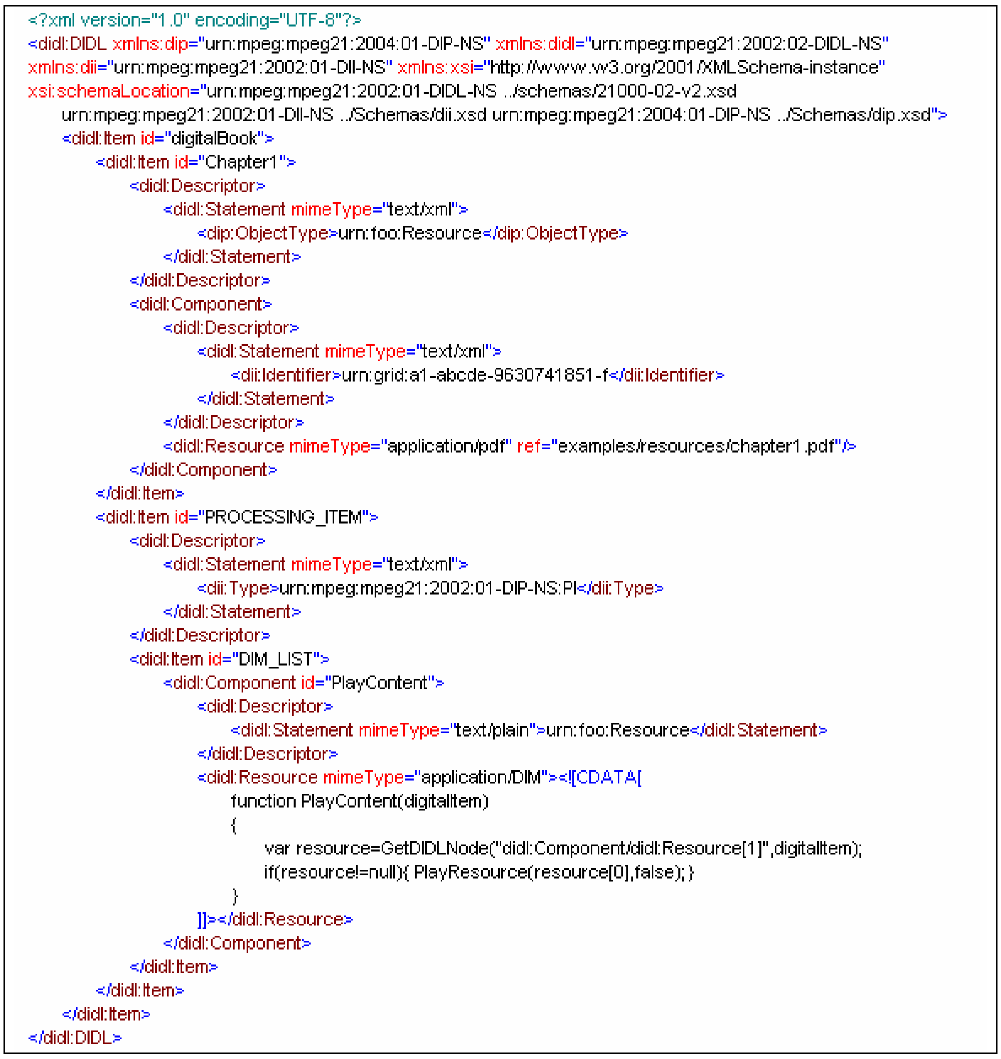
Table 5 gives the usual but not normative prefix and the corresponding namespace:

Table 5. DIP prefix and namespace.

Dip	urn:mpeg:mpeg21:2005:01-DIP-NS

The DIML specification normatively includes the ECMAScript Language Specification detailed in ECMA-327 (edition 3) [42]. This provides a standardised core language specification for DIML, including specification of the following features of DIML: lexical conventions and syntactical structure, flow control structures (i.e. if/then, while, etc.), primitive data types (String, Boolean and Integer), composite data types (Objects and arrays), standard arithmetic, logical and bitwise operators, exception handling, error definitions ad support for Regular Expressions.

Moreover, the DIML specification includes a normative set of DIML specific object types and global object properties. The DIML object types are specified by the object properties (including primitive values, other objects, and functions), and attributes of those properties. The DIML global object properties is a normative set of host-defined properties of the global object.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<didl:DIDL xmlns:dip="urn:mpeg:mpeg21:2004:01-DIP-NS" xmlns:didl="urn:mpeg:mpeg21:2002:02-DIDL-NS"
xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2002:01-DIDL-NS ../schemas/21000-02-v2.xsd
    urn:mpeg:mpeg21:2002:01-DII-NS ../Schemas/dii.xsd urn:mpeg:mpeg21:2004:01-DIP-NS ../Schemas/dip.xsd">
    <didl:Item id="digitalBook">
        <didl:Item id="Chapter1">
            <didl:Descriptor>
                <didl:Statement mimeType="text/xml">
                    <dip:ObjectType>urn:foo:Resource</dip:ObjectType>
                </didl:Statement>
            </didl:Descriptor>
            <didl:Component>
                <didl:Descriptor>
                    <didl:Statement mimeType="text/xml">
                        <dii:Identifier>urn:grid:a1-abcde-9630741851-f</dii:Identifier>
                    </didl:Statement>
                </didl:Descriptor>
                <didl:Resource mimeType="application/pdf" ref="examples/resources/chapter1.pdf"/>
            </didl:Component>
        </didl:Item>
        <didl:Item id="PROCESSING_ITEM">
            <didl:Descriptor>
                <didl:Statement mimeType="text/xml">
                    <dii:Type>urn:mpeg:mpeg21:2002:01-DIP-NS:PI</dii:Type>
                </didl:Statement>
            </didl:Descriptor>
            <didl:Item id="DIM_LIST">
                <didl:Component id="PlayContent">
                    <didl:Descriptor>
                        <didl:Statement mimeType="text/plain">urn:foo:Resource</didl:Statement>
                    </didl:Descriptor>
                    <didl:Resource mimeType="application/DIM"><![CDATA[
                        function PlayContent(digitalItem)
                        {
                            var resource=GetDIDLNode("didl:Component/didl:Resource[1]",digitalItem);
                            if(resource!=null){ PlayResource(resource[0],false); }
                        }
                    ]]></didl:Resource>
                </didl:Component>
            </didl:Item>
        </didl:Item>
    </didl:Item>
</didl:DIDL>
```

Figure 10. Digital book with a single chapter and DIP information.

The DIM *declaration* refers to the declaring of the method as being part of a particular Digital Item. The DIM *definition* refers to code written in the DIML that defines the method. The method definition may be listed in a separate DIM location and referenced from the DID, or it may be embedded inline in the DID. In either case it is the method definition itself that is the *resource* (in terms of the Digital Item Declaration Model).

The different possibilities for the definition of a DIM are the following: DIM embedded as base64 encoded data, DIM embedded in a CDATA section, DIM Referenced.

Figure 10 shows an example of how a DIM can be embedded in a CDATA section in a DIDL document.

Figure 10 represents a digital book which has a single chapter. A PlayContent DIM that opens a chapter of the book (in the current case, it is restricted to a single chapter) is embedded in a CDATA section of the DIDL document. The DIM takes as input a Resource of type urn:foo:Resource, as specified in the Descriptor element of the DIM. This means that this DIM can have as input any node of the current DIDL document (or even also from external DIDL documents) that is identified to be of type urn:foo:Resource, by means of the ObjectType element. In the above example, Chapter1 Item is identified to be of that type, what means that the item node is a possible candidate to be the input for the item. In the above case, it is upon the application to make the user decide which is the resource that he wants to play, among the available ones. Another possibility would be to include directly in the DIM the necessary steps to make the user select between the available resources. In the latter case, the application would only need to interpret the DIM code.

2.3 OMA DRM

OMA DRM (Open Mobile Alliance DRM) [7] has been developed to enable the controlled consumption of digital media objects by allowing content providers the ability, for example, to manage previews of DRM Content, to enable super distribution of DRM Content, and to enable transfer of content between DRM Agents.

OMA DRM defines a set of Actors and Components in its reference architecture. The most relevant are the DRM Agent, Content Issuer, Rights Issuer, User and Off-device Storage.

2.3.1 OMA DRM architecture

The **DRM Agent** (DRM-A) represents a trusted entity in a device. This entity enforces permissions and constraints associated with DRM content, controlling the access and usage of DRM content.

The **Content Issuer** (CI) delivers DRM content. OMA DRM defines the DRM content format to be delivered to DRM-A, and also defines the way the content can be transported from a CI to a DRM-A using alternative transport mechanisms. The DRM content packaging may be handled directly by the CI or it may receive it from an external source.

The **Rights Issuer** (RI) is the OMA DRM entity that assigns permissions and constraints to DRM content, and generates Rights Objects (RO). The RO is represented in XML and expresses permissions and constraints associated with DRM content.

A **User** is a human user of DRM content, which can only access DRM content through a trusted DRM Agent.

The **Off-Device Storage** allows DRM content to be stored off-device, for backup purposes, to free memory on the device. RO with stateless permissions can also be off-device stored.

The OMA DRM system enables CI to distribute Protected Content and RI to issue Rights Objects for the Protected Content. The DRM system is independent of media object formats, operating systems, and runtime environments. For User consumption of the Content, Users acquire Permissions to Protected Content by contacting RI. RI grants appropriate Permissions for the Protected Content to User Devices. The Content is cryptographically protected when distributed; hence, Protected Content will not be usable without an associated Rights Object issued and cryptographically bound to the User's Device.

The Protected Content can be delivered to the Device by any means. But the Rights Objects are tightly controlled and distributed by the RI in a controlled manner. The Protected Content and Rights Objects can be delivered to the Device together, or separately. The system does not imply any order or “bundling” of these two objects. It is not within the scope of the DRM system to address the specific payment methods employed by the RI.

The OMA DRM specifications define the format and the protection mechanism for DRM Content, the format and the protection mechanism for the Rights Object, and the security model for management of encryption keys. The OMA DRM specifications also define how DRM Content and Rights Objects may be transported to devices using a range of transport mechanisms. Any interaction between network entities is out of scope.

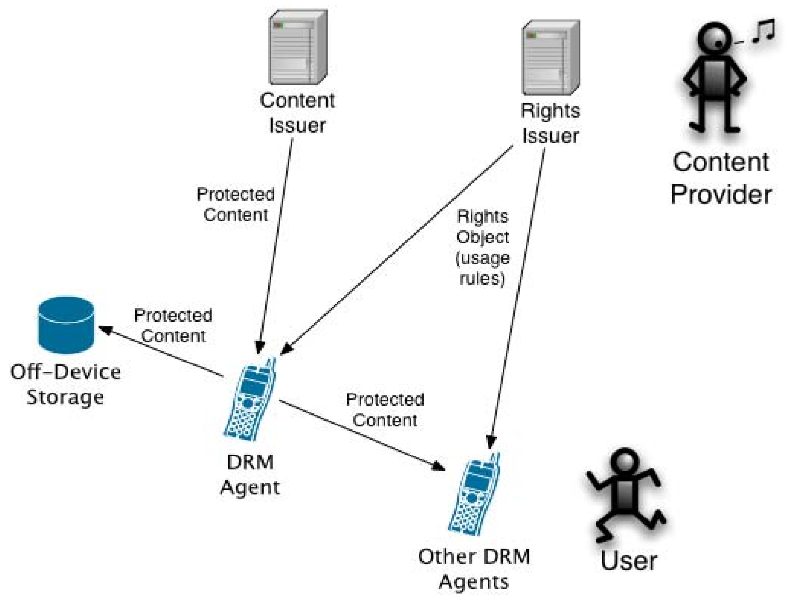


Figure 11. OMA-DRM generic architecture.

The basic steps for distributing DRM Content can be summarised as follows:

1. Content packaging: Content is packaged in a secure content container (DCF). DRM Content is encrypted with a symmetric content encryption key (CEK). Content can be pre-packaged, i.e. content packaging does not have to happen on the fly. Although not required by the OMA DRM specifications or the OMA DRM architecture, it is recommended that the same CEK is not used for all instances of a piece of content. Using the same CEK for all content instances would pose a greater risk if a single device was to be hacked and a CEK stored on that device exposed. Using a different CEK for different deliveries or different devices will limit this risk.

2. DRM Agent authentication: All DRM Agents have a unique private/public key pair and a certificate. The certificate includes additional information, such as maker, device type, software version, serial numbers, etc. This allows the content and rights issuers to securely authenticate a DRM Agent. Any privacy aspects with releasing such information are addressed in the technical specifications.
3. Rights Object generation: A Rights Object is an XML document, expressing the permissions and constraints associated with the content. The Rights Object also contains the CEK – this ensures that DRM Content cannot be used without an associated Rights Object.
4. Rights Object protection: Before delivering the Rights Object, sensitive parts are encrypted (e.g. the CEK), and the Rights Object is then cryptographically bound to the target DRM Agent. This ensures that only the target DRM Agent can access the Rights Object and thus the DRM Content In addition, the RI digitally signs the RO.
5. Delivery: The RO and DCF can now be delivered to the target DRM Agent. Since both are inherently secure, they can be delivered using any transport mechanism (e.g. HTTP/WSP, WAP Push, MMS). They can be delivered together, e.g. in a MIME multipart response, or they can be delivered separately.

The DRM Agent has to be trusted by the rights issuer, both in terms of correct behaviour and in terms of a secure implementation. In OMA DRM, each DRM Agent is provisioned with a unique key pair, and an associated certificate, identifying the DRM Agent and certifying the binding between the agent and this key pair. This allows rights issuers to securely authenticate the DRM Agent using standard PKI procedures.

The information in the certificate enables the Rights Issuer to apply a policy based on its business rules, the value of its content, etc. For example, a rights issuer may trust certain manufacturers, or it may keep an updated list of DRM Agents that are known to be good or bad according to some criteria defined by the rights issuer. It is also possible for a group of stakeholders to establish a joint authority identifying trusted DRM Agents, with legally binding compliance rules.

Revocation in this model amounts to not distributing content any more to DRM Agents that are no longer considered trusted.

What constitutes a trusted DRM Agent depends on the policy and business model of rights issuers. For example, if a hack or a fault compromises a whole class of devices, a rights issuer may decide to stop distributing new content to all devices of that type or class. This is a worst-case scenario. At the other end of the spectrum, maybe there is a known bug in devices of a certain type, but the risk of content leaking is relatively small. In such cases, content and rights issuers may choose to continue to deliver content to existing devices, and instead let manufacturers correct the problems in future versions. Anyway, the secure mechanism for authenticating DRM Agents enables rights issuers to enforce such policies.

The DRM Content Format (DCF) is a secure content package for encrypted content, with its own MIME content type. In addition to the encrypted content it contains additional information, such as content description (original content type, vendor, version, etc.), rights issuer URI (a location where a Rights Object may be obtained), and so on. This additional information is not encrypted and may be presented to the user before a Rights Object is retrieved. Since a DCF is inherently secure, it can be transported using any transport protocol, e.g. in an HTTP response or in an MMS message. It can be stored for back-up on any kind of storage, e.g. removable media or a networked PC. It can be copied and sent to another DRM Agent, where a Rights Object may be acquired for use on the receiving device (superdistribution). The content encryption key needed to unlock DRM Content inside a DCF is contained within a Rights Object. Thus it is not possible to access DRM Content without a Rights Object. DRM Content can only be used as specified in a Rights Object. OMA DRM includes a mechanism allowing a DRM Agent to verify the integrity of a DCF, protecting against modification of the content by some unauthorised entity.

Rights Objects are used to specify consumption rules for DRM Content. The Rights Expression Language (REL) defined by OMA DRM specifies the syntax (XML) and semantics of permissions and constraints governing the usage of DRM Content. An instance of a rights document is called a Rights Object, and has its own MIME content type. Rights Objects are made up of permissions (e.g. play, display and execute) and constraints (e.g. play for a month, display ten times). Rights Objects may also include constraints that require a certain user (user identity) to be present when the content is used. These permissions and constraints, along with other information embodied in the Rights Object, (e.g. copyright information) may be presented to the user. The Rights Object also governs access to DRM Content by including the content encryption key (CEK).

A single Rights Object may be associated with multiple pieces of DRM Content. Further, it is possible to assign different permissions to different components of a composite object.

Conversely, a single piece of DRM Content may be associated with multiple Rights Objects. If there are multiple Rights Objects associated with a piece of DRM Content, each Rights Object is treated individually – Rights Objects are not combined. This means that at any one time, there may be more than one Rights Object whose constraints are satisfied. When this is the case, the DRM Agent selects one to enforce. This selection may be made automatically by the DRM Agent based on some selection criteria, e.g. picking the least restrictive Rights Object, or it may be done based on user interaction.

A Rights Object is protected using a rights encryption key (REK). The REK is used to encrypt sensitive parts of the Rights Object, such as the CEK. In addition, the RO is digitally signed by the RI.

During delivery, the REK is cryptographically bound to the target DRM Agent. In this way only the target DRM Agent can access the Rights Object, and thus the CEK.

Since a protected Rights Object is inherently secure, it can be copied and stored off-device for backup purposes. Some permissions require maintenance of state by the DRM Agent, for example a limited number of plays. Rights Objects containing such permissions cannot be copied or stored off-device, if this would result in loss of state information - e.g. current number of plays.

2.3.2 The Rights Object Acquisition Protocol (ROAP) Suite

The Rights Object Acquisition Protocol (ROAP) is the common name for a suite of DRM security protocols between a Rights Issuer (RI) and a DRM Agent in a Device. The protocol suite contains:

- 4-pass protocol for registration of a Device with an RI and two protocols by which the Device requests and acquires Rights Objects (RO). The Registration protocol is a complete security information exchange and handshake between the RI and the

Device and is generally only executed at first contact, but may also be executed when there is a need to update the exchanged security information, or when DRM Time in the Device is deemed inaccurate by the Rights Issuer. This protocol includes negotiation of protocol parameters and protocol version, cryptographic algorithms, exchange of certificate preferences, optional exchange of certificates, mutual authentication of Device and RI, integrity protection of protocol messages and optional Device DRM Time synchronisation. Successful completion of the Registration protocol results in the establishment of an RI Context in the Device containing RI-specific security related information such as agreed protocol parameters, protocol version, and certificate preferences. An RI Context is necessary for execution of the other protocols in the ROAP suite.

- 2-pass RO acquisition protocol encompasses request and delivery of an RO. It is the protocol by which the Device acquires Rights Objects. This protocol includes mutual authentication of Device and RI, integrity-protected request and delivery of ROs, and the secure transfer of cryptographic keying material necessary to process the RO. The successful execution of this protocol assumes the Device to have a pre-established RI Context with the RI.
- 1-pass RO acquisition protocol is only a delivery of an RO from an RI to a Device (e.g. messaging/push). Its successful execution assumes the Device to have an existing RI Context with the sending RI. In contrast to the 2-pass RO acquisition protocol, it is initiated unilaterally by the RI and requires no messages to be sent by the Device. One use case is distribution of Rights Objects at regular intervals, e.g. supporting a content subscription. The 1-pass protocol is essentially the last message of the 2-pass variant.
- 2-pass Join Domain protocol is the protocol by which a Device joins a Domain. The protocol assumes an existing RI Context with the RI administering the Domain. Successful completion of the Join Domain protocol results in the establishment of a Domain Context in the Device containing Domain-specific security related information including a Domain Key. A Domain Context is necessary for the Device to be able to install and utilise Domain ROs.
- 2-pass Leave Domain protocol is the protocol by which a Device leaves a Domain. The protocol assumes an existing RI Context with the RI administering the Domain.

2.3.3 Rights Expression Language

OMA (Open Mobile Alliance) has developed the OMA DRM Rights Expression Language versions [43] based on ODRL [44].

Rights are the collection of permissions and constraints defining under which circumstances access is granted to DRM Content. The structure of the rights expression language enables the following functionality:

1. Metadata such as version and content ID
2. The actual rights specification consisting of
    a. Linking to and providing protection information for the content, and
    b. Specification of usage rights and constraints

Models are used to group rights elements according to their functionality, and thus enable concise definition of elements and their semantics. The following models are used throughout this specification:

- Foundation model: constitutes the basis for rights. It contains the rights element bringing together meta-information and agreement information. The foundation model serves as the starting point for incorporating the agreement model and the context model.
- Agreement model: expresses the Rights that are granted over a DRM Content. It consists of the agreement element connecting a set of Rights with the corresponding DRM Content specified with the asset element. The agreement model incorporates the permission model and the security model
- Context model: provides Meta information about the rights. It augments the foundation model, the agreement model, and the constraint model by expressing additional information.
- Permission model: augments the agreement model. It facilitates the expression of permissions over assets by specifying the access granted to a device. The permission model incorporates the constraint model allowing fine-grained consumption control of DRM Content. The set of permissions comprises play, display, execute, print, and export. The usage of the DRM Content MUST be only granted according to the permissions explicitly specified by the corresponding Rights Object(s). A permission that does not contain a constraint child element is unconstrained and access according to the respective permission element(s) MUST be granted. Note that the REL only specifies consumption and export rights and not management rights, e.g., install, uninstall, delete, or distribution rights. This is made possible by the separation of DRM Content and Rights Objects (although DRM Content and Rights Objects may be delivered together) freeing the REL from unnecessary complexity and overhead. Content can be stored; however, it can only be accessed if a corresponding Rights Object is available. Similarly, encrypted content can be super-distributed without unnecessarily complicating the REL; no separate distribution permissions are necessary, since DRM Content without the decryption key is of no value. The DRM Agent MUST ignore unknown or unsupported permission elements. The DRM Agent MUST NOT grant alternative, not explicitly specified rights to access Content instead. Known and supported permission elements defined by the same Rights Object MUST remain unaffected and the DRM Agent MUST grant access according to those. A Permission that is not granted due to unknown or unsupported constraints (section 5.5) MUST NOT affect the granting of other permissions.
- Constraint model: enhances the permission model by providing fine-grained consumption control of content. Constraints are associated with one permission element at a time. For a permission to be granted all its constraints MUST be fulfilled. If a constraint is not understood or cannot be enforced by the consuming device the parent permission is invalid and must no be granted. If present, a constraint element should contain at least one of its child elements. If a constraint element does not contain any constraints such as count, datetime, etc. it is unconstrained, and a DRM Agent must grant unconstrained access according to the permission containing such an unconstrained constraint element.

- Inheritance model: describes how a parent Rights Object can specify Permissions and Constraints for one or more pieces of DRM Content each governed by a child Rights Object, using a limited subset of the ODRL inheritance model. The DRM Agent must not accept parent child Rights Objects constellations with more than one level of inheritance (i.e. parent-child). In other words, a parent Rights Object must not inherit Permissions and Constraints from another Rights Object.
- Security model: Security constitutes an important part of a DRM system. OMA DRM 2.0 provides confidentiality for the CEK of Rights Objects, integrity of the association between Rights Objects and DRM Content and Rights Object integrity and authenticity. The ODRL security model, which forms the basis for the security model of this specification, is based on XMLENC [45] and XMLSIG [46].

2.4 DReaM

DReaM [8] is a Sun initiative to develop a DRM solution focusing on open-standards. According to DReaM own information, whenever the market requires proprietary solutions DReaM will be capable of integrating with these solutions providing openness and interoperability that meets customer requirements. DReaM is an initiative to leverage the methodology of Service Oriented Architectures (SOA) and introduce rights management services that leverage open standards and support cross-service capabilities.

The DReaM architecture supports the separation between the rights management components through the decoupling of authentication, licensing, rights management and protection systems. This disintermediation enables the choice and selection of these technologies independent of each other without any compromise for the overall solution. There are two key elements for disintermediation in DReaM:

- Separation of rights management from the content protection systems.
- Separation of identity and authentication services from individual hardware devices.

DReaM has a central objective towards the creation of an interoperable DRM, offering the capability to interoperate directly with other content protection technologies and supporting services that enable both Conditional Access System (CAS) and DRM. A key-concept in the DReaM platform is the disintermediation concept – this enables multiple instances of these components to exist in a DRM/CAS system. The DReaM disintermediation system (see Figure 12) enables the coexistence of multiple instances of content protection specific components (player, licensor and packager) and components that are not content protection specific (disintermediation agent, conductor, catcher, licensing conductor, contracts manager, authentication service, shop and transaction system and content delivery system).

The process of disintermediation happens as follows:

1. Client requests a license
2. Front-end service redirects client to a client disintermediation agent
3. Disintermediating agent contacts Conductor (back-end service)
4. Conductor contacts back-end services for authentication and rights verification
5. Conductor signals front-end service with instructions to deliver license to client
6. Front-end service delivers license

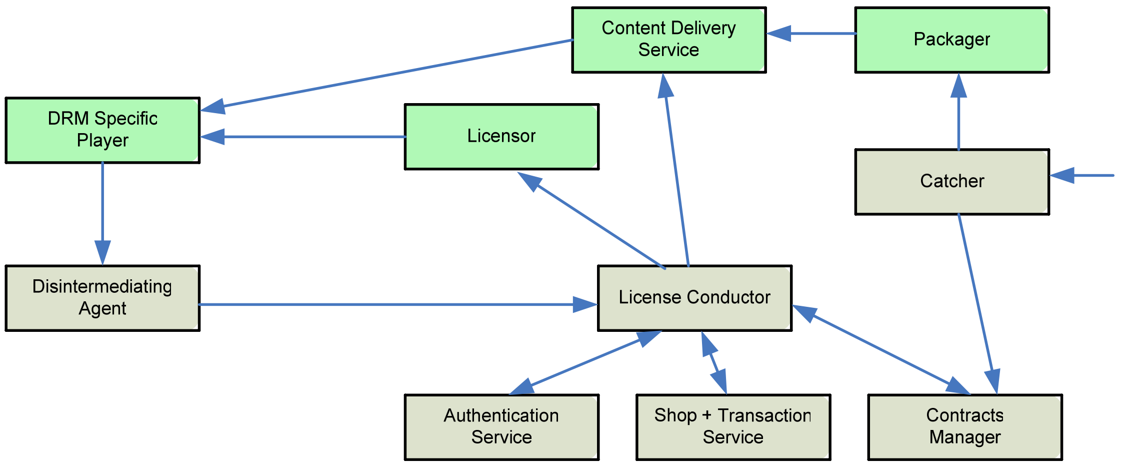


Figure 12. The DReaM DRM disintermediation system.

Although DReaM's objective is to offer the capability to interoperate directly with other content protection technologies, it has developed a new method of expressing and controlling rights for content, which is termed DreaM-MMI (Mother-May-I).

DReaM has produced for the moment the DReaM-CAS and DReaM-MMI specifications, which are available upon request.

According to Sun, the design philosophy underlying DReaM-MMI is that clients should be able to negotiate for rights through standardised protocols rather than downloading a license with an embedded expression of rights. Access to content is requested under certain conditions, and the client software manages the use, according to the guidelines under which the content is requested. It is done in the following manner:

- A DReaM-MMI compliant client will request the use of given protected content under a specific set of usage terms (e.g. number of viewings, etc.)
- The DReaM-Licensor responds to the client's DReaM-MMI request after communicating with the DReaM Contracts Manager to determine whether the client should be allowed access to the content on those DReaM-MMI expressed terms.
- If the Licensor response is positive, the content keys are delivered to the client where it will be consumed according to the terms expressed in the DReaM-MMI request.
- The DReaM-MMI compliant client has the responsibility for enforcing that the content is only used under those specified terms.

If a client wishes to access content under different usage terms, the client could renegotiate with the DReaM-Licensor. No more access is allowed than the specific rights the client had requested.

An important issue in DreaM-MMI is that no Rights Expression Language (REL) is delivered to the client. This approach for the rights management can be seen as a Sun intend to avoid the patent issue regarding Rights Expression Languages.

DReaM defines a set of Actors and Components in its architecture. The most relevant are the following:

- Client - DRM Specific Player: It is a client-side player application that has DRM specific support for handling protected content and licenses.
- Client - Disintermediating Agent: It is a Java application that would perform the re-direction required for disintermediation.
- Licensor: The licensor is tightly bound to the DRM specific content protection technology.
- Licensing Conductor: It the role of managing the licensing processes involved in the DReaM solution. It has interfaces to the DReaM Client, Shopping and Transaction Service, Authentication Service, Contracts Manager and the Licensor. It performs the necessary e-commerce transactions and authentication of the user. It instructs the Licensor to generate the license for a given user for specific content.
- Contracts Manager: It stores business rules associated with content, as well as user rights. This component has interfaces to the Licensing Conductor and the Licensor. The Licensor will generate a license for a given piece of content based on the business rules and user rights that are available in the Contracts Manager.
- Authentication Service: It is where subscribers, users and devices are cleared for access to services and content. The methods of authentication vary from weak methods such as username + password challenge to stronger authentications such as smart cards or biometrics.
- Shop and Transaction Service (Business Support Services): The work flow functions of shopping and transacting purchases includes everything from collecting payments from buyers to paying sellers and making sure that everyone is appropriately compensated in a secure manner.
- Content Delivery Server: The content distribution server will receive protected content from the packager. Stream keys used for content protection in the packager may be optionally stored with the content in the content delivery server.
- Packager: The packaging process involves combining content data/files with associated metadata and creating logical packages that include the defined business rules. DRM packaging applications may have user interfaces for the human processing of content or the rights may be machine processed from business rules that are made available at the time of content ingestion. These business rules may be stored in a content management system (CMS), and the DRM packager would then read them through database queries.
- Catcher: The Catcher performs content ingestion. It receives content and associated business rules from the content supplier. The content, which is unprotected at this stage, is passed to the Packager. The business rules associated with the content are passed to the Contracts Manager.

2.5 Marlin

The aim of Marlin [9] is to create a DRM system that interoperates among devices from different vendors.

It is based on previous work developed by Intertrust in Nemo and Octopus projects. Whereas the first is a secure messaging architecture based on web services for digital media distribution and rights management, the latter is a software toolkit for developing lightweight DRM systems based on elementary graph theory.

In Octopus, we have nodes for entities in a DRM scheme that represent users, domains, devices and subscriptions (licenses).

A device has rights over a content governed by a subscription if there is a series of links that connect the device to a subscription through a user. Those links are created by e-commerce systems which are Marlin compliant.

To determine if a user has rights over a content, there must be a series of links that connect the user to a subscription. The subscription node points to a content object which contains on one hand a control program written in a bytecode language called Plankton and the keys used to decrypt the content. In this way, when a user wants to exercise a right over a content, the Marlin client will run the control program associated to that content to determine whether he is authorised or not by checking if there are existing links from the client node to the user's identity.

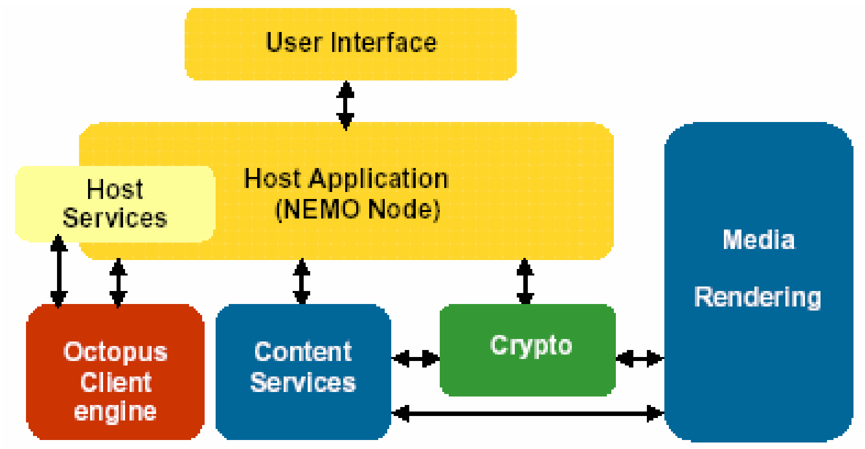


Figure 13. Octopus DRM Client System Elements. Source: [10]

An important aspect of Marlin is that it avoids the usage of Rights Expressions Languages (RELs) by avoiding a descriptive grammar, and thus, the patent issue regarding Rights Expression Languages.

Marlin includes an OMA Gateway, which enables Marlin Clients to act as OMA DRM Agents. The combination of the Marlin OMA Gateway with a Marlin DRM Client will satisfy all requirements for an OMA DRM Agent, and therefore, can be considered to be an OMA DRM Agent in all respects. For this reason, OMA content can be received, processed, and used as it would on any other OMA DRM compliant device. This means that no modifications are required on the part of OMA Rights Issuers to interact with "Marlin-based" OMA DRM agents. For example, Rights issuers will not need to add additional permissions nor would they have to modify the protocols used to communicate with Marlin-based DRM Agents.

Marlin defines several Actors in its architecture:

- User: A user is an individual that interacts with Service Providers to acquire licenses for digital content, and interacts with Marlin Clients to access or manage use of that content. In most instances, users are also "License Owners", that is, they have purchased the required rights to use the content under given circumstances. In some use cases however, such as in "sharing" or "superdistribution" cases, the user does not have a license to use the content and must interact with a Service Provider or a valid License Owner to gain access to the content.

- Service Provider: A Service Provider is the generic term used to describe the entity or organisation that is responsible for selling or distributing digital media content and associated licenses. Service Providers are not limited to any particular type of business model for licensing or distributing content. They can choose to support "a-la-carte" individual content licenses, rentals, subscription-based services, or a hybrid of these.
- Content Provider: A Service Provider may serve the role of a Content Provider and aggregate content from Content Owners and distribute it to Users. However, Marlin is flexible and allows for a variety of mechanisms for content delivery. For example, content may be delivered in peer-to-peer fashion or from a Broadcaster via a broadcast channel.

Marlin defines several modular components that are designed to serve a particular purpose or role within the system. Its main components are the following:

- Marlin Client: It is responsible for requesting licenses and links, and controlling access to protected content. A Marlin Client may be implemented in a hardware device (such as a portable media player) or as a client application (such as a PC software media player application). When the host device (or player application) requests access to content, the Marlin Client will execute the control program in the license and check for the presence of any required links. Then, if permitted by the license, the Marlin client will allow the content key to be decrypted and used to access the protected content.
- Domain Manager: It serves the function of creating a domain and managing the devices and users that are associated with the domain. To do this, a domain manager issues Device-Domain Link Objects that associate devices to a domain, and Domain-User Link Objects that associate the domain to users that 'own' the domain. A domain manager can either be operated by a Service Provider at a remote location accessible via the internet, or can be implemented on a Marlin device in the user's local network. The 'rules' that a Domain Manager must use to administer the domain are defined via a Domain Policy.
- Registration Service: It is typically operated by a Service Provider, and serves the role of issuing Nodes and Links. During the purchase process (or a subscription renewal process), the Marlin Client receives a trigger to contact a Registration Server to request the appropriate nodes and/or links to support the transaction. For example, when purchasing content a Marlin Client would be instructed to request a User Node corresponding to the user, and a link from the Client to that User. Alternatively, in a subscription renewal, the Marlin client might be automatically triggered to request a new User Subscription link from the Registration Server (e.g., one with an extended expiration date).
- License Service: A Service Provider also operates a License Service that is responsible for issuing licenses to Marlin Clients. After a service provider's e-commerce system processes a transaction for a purchase, the ecommerce system will trigger the Marlin Client to contact a License Service with a request for a license. The License Service will generate the necessary license objects and respond with a "License Bundle" that the Marlin Client will use to govern access to the content.

Additional functional components are also defined in Marlin. These other components serve functions such as ensuring that Clients have the most current versions of security metadata, trusted time values, etc.

2.6 OpenSDRM

OpenSDRM [11] is an adaptable DRM architecture since it can be configured for use with several business models and different types of content. OpenSDRM deploys a traditional DRM solution for content rights protection and can be applied for publishing and trading of digital multimedia content.

The proposed security architecture started from the OPIMA international specifications, MPEG-4 IPMP Extensions and the emerging MPEG-21 IPMP architecture as well as with some of the proposals for JPEG2000 standard Part 8 – JPSEC – JPEG2000 security. OpenSDRM was developed primarily in the scope of the MOSES project. MOSES was an FP5 EC project joining some companies over Europe that is implementing the new MPEG IPMP Extensions framework and at the same time developing business models and applications for secure content exchange between embedded devices. This DRM solution is composed of several optional elements covering the content distribution value chain, from content production to content usage. It covers several major aspects of the content distribution and trading: content production, preparation and registration, content, interactive content distribution, content negotiation and acquisition, strong actors and user's authentication and conditional visualisation/playback. Even though the MOSES project refers explicitly to MPEG-4 file format as the content format, this infrastructure was designed with the concern to be adaptable and applicable to all types of content and business models (both for download, streaming or even broadcasting).

OpenSDRM uses two important concepts largely referenced afterwards: Actors and Components. An Actor is a person, an entity or an organisation that uses a Component. A Component is a tool cooperating to offer a set of DRM-related functionalities. Its architectural elements are described hereafter:

- Authentication Server (AUS): It is used for Actors and Components registration and authentication. Its main functions are: (a) components registration in the system, including functionalities to update and to delete/revoke Components; (b) users registration that will interact with the Components. It is also used to update and delete/revoke users; (c) verify if a user has or not a valid wallet installed on its system; and (d) available payment gateways verification and validation.
- Configuration Server (CFS): It registers the location of all the Components, so that they can be accessed from other components. The main functionalities of this server are: (a) component location registration and its details; and (b) specific component information provision to other components.
- License Server (LIS): It creates, manages and delivers licenses. The main functionalities of this server are: (a) content protection keys secure storage; (b) user available licenses reporting; (c) specific user and content identifier license creation; (d) update and delete/revoke licenses on the system; and (e) license retrieval by end users systems. This mechanism bounds the license together with the content key(s) in a protected manner and sends them back to the end-user.
- Protection Tools Server (ITS): It registers the new protection tools and receives authenticated requests for specific protection tool downloading. The server makes protection tools available to the CPS to allow the content protection. Major functionalities include: (a) new protection tools introduction; (b) available protection tools listing; (c) check the provided protection functionalities; and (d) available protection tools download.

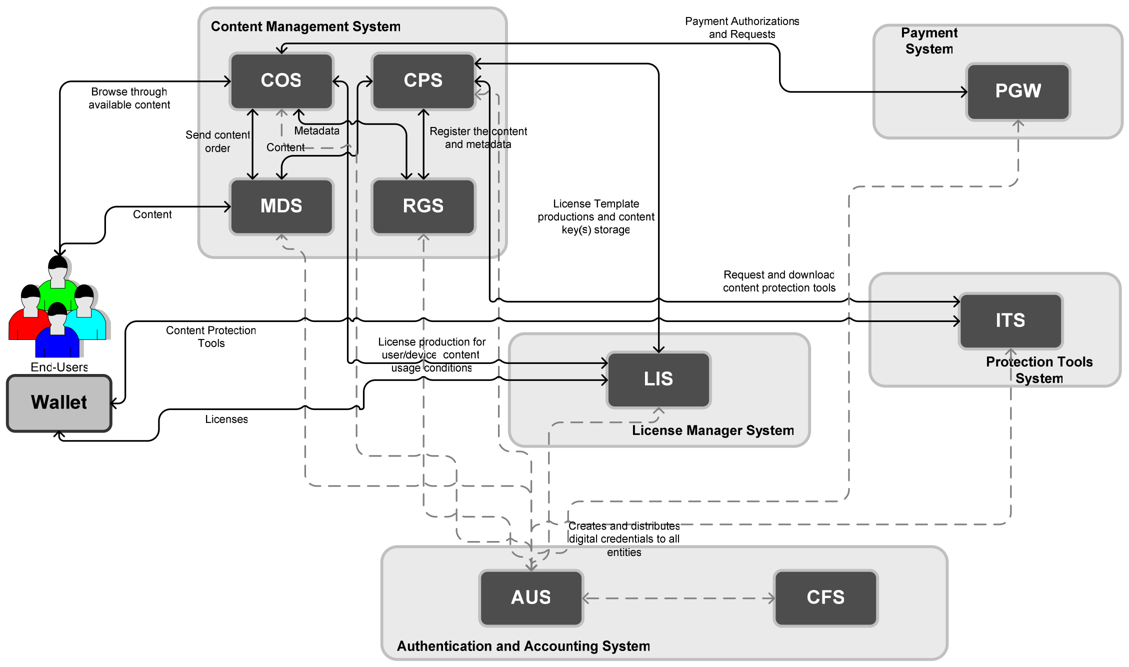

Content Management System
Payment Authorizations and Requests
Payment System
PGW
COS
CPS
Browse through available content
Send content order
Metadata
Content
Register the content and metadata
MDS
RGS
Content
License Template productions and content key(s) storage
Request and download content protection tools
ITS
Protection Tools System
Content Protection Tools
License production for user/device content usage conditions
LIS
End-Users
Wallet
Licenses
License Manager System
Creates and distributes digital credentials to all entities
AUS
CFS
Authentication and Accounting System

Figure 14. OpenSDRM DRM platform architecture. Source: [76]

- Payment Gateway Server (PGW): It verifies and validates the payment methods provided by a User. Its major functions are: (a) request payment clearance from the billing system, clearing the user's payment instrument; (b) payment capture from a previously cleared transaction; and (c) subscription of new content stores to the billing system.
- Content Preparation Server (CPS): It receives from a source and encodes raw content to a specific format, adds metadata and protects it according to a protection tool(s). The encoding format is out of the scope of OpenSDRM. The Registration Server (RGS) assigns unique identifiers to content. The main functionalities are: (a) new content registration; (b) metadata registration; (c) list the available content and metadata on the system that matches with specific criteria. Media Delivery Server (MDS) registers the storage and delivery of content to the client. Main functions include: (a) store content location on the system; (b) notify the system of content user requests; and (c) notify the system of content downloads. Commerce Server (COS) is responsible for presenting and trading content with the users.
- Wallet: It is a client-side component that interacts with the OpenSDRM platform to enforce the DRM functions on owned DRM-governed content. Its main functions include: (a) user registration and certification; (b) wallet validation and certification; (c) sensitive information secure storage; (d) download of licenses whenever necessary; and (e) download of protection tools.
- Content Rendering Application (Media Player) component: It renders protected and governed content. The Media Player is connected to the Wallet to access information needed to obtain the necessary license(s) and key(s) to render the content. Its implementation is dependent on the type of content that is protected, while the DRM mechanisms are not.

License Management in OpenSDRM

One of the interesting mechanisms provided by OpenSDRM is the fact that licenses are handled at the client-side by a middleware layer, called OpenSDRM Wallet. This wallet is capable of managing the access to protected content by different Applications. Every time an application wishes to perform an operation over the content, it contacts the wallet that authorises or not such operation according to what is specified on the license. This layer allows the coexistence of many DRM-protected files and DRM-enabled applications on a single client system, presenting a horizontal approach to DRM. Nowadays DRM approaches are vertical: examples include Microsoft Windows Media Rights Management or iTunes. While a solution like Microsoft Windows Rights Management is end-to-end Microsoft system-dependent (even at the client-side), relying on Windows Media Player to obtain the licenses and enforce it to the content, OpenSDRM follows a more horizontal approach in which several content applications can share the access to content, mediated by the OpenSDRM Wallet. This represents in fact an important interoperability layer at the client-side.

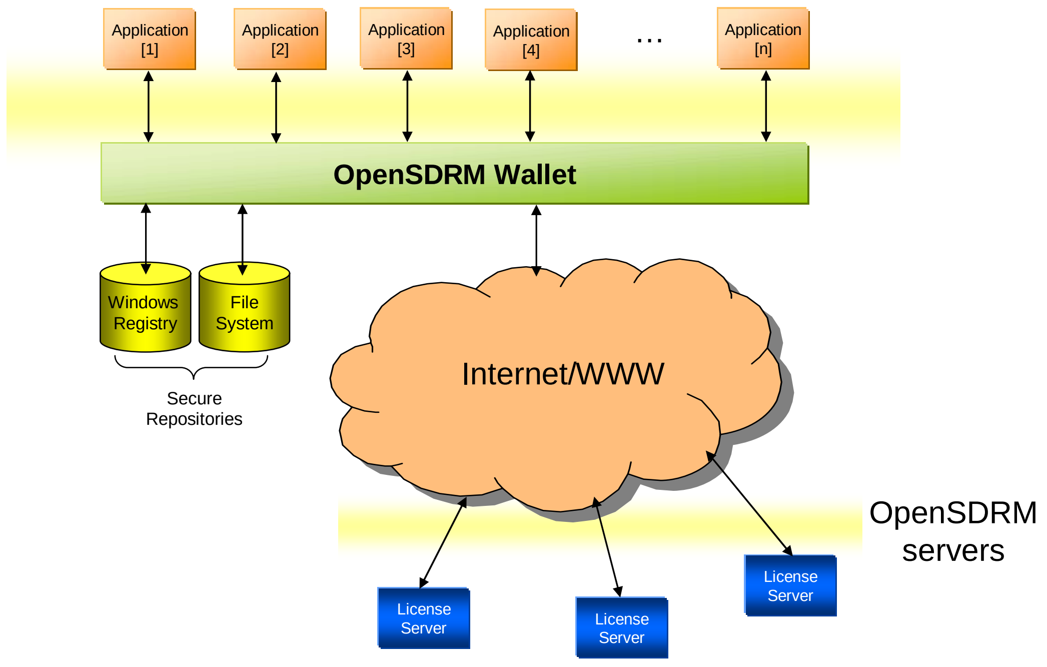


Figure 15. OpenSDRM Wallet mediating Access to licenses. Source: [65]

A previous and important step is executed between the content application and the OpenSDRM wallet in order to authenticate the application so that it can request content operations to the wallet (this may include receiving content deciphering keys provided in the ODRL licenses). This means that any of the applications that wish to use this system will need to know how to execute the following two processes:

- Enrol to and request authentication to the OpenSDRM wallet, exchanging a set of credentials with the OpenSDRM Wallet, to enable application authentication and the establishment of a secure channel between the application and the wallet;
- Request authorisation to the OpenSDRM wallet to perform operations over the content. This process includes the extraction of content unique identifier and requesting the wallet the permission to use the content. The wallet is responsible for getting the license from the server, parsing it, analysing the rights that are associated to it before the allowance or rejection of the operation over the content (this may

include passing the decryption key to the application or the appropriate protection tool).

On the server-side of the OpenSDRM solution, one or more License servers can issue licenses (ODRL formatted) that are bound to the user and content. These licenses specify how content can be used by the user.

2.7 Windows Media Rights Manager

According to Microsoft, DRM or “Digital Rights Management” is the process of protecting digital media controlling its usage by licenses, which confer specific rights to the end-user. The digital media is encrypted using a key that blocks/unblocks the usage of the latter. After the encrypting process, the media can only be played using a license (emitted by a License provider) containing the key that unlocks it.

The following image represents the Microsoft’s DRM [12] process.

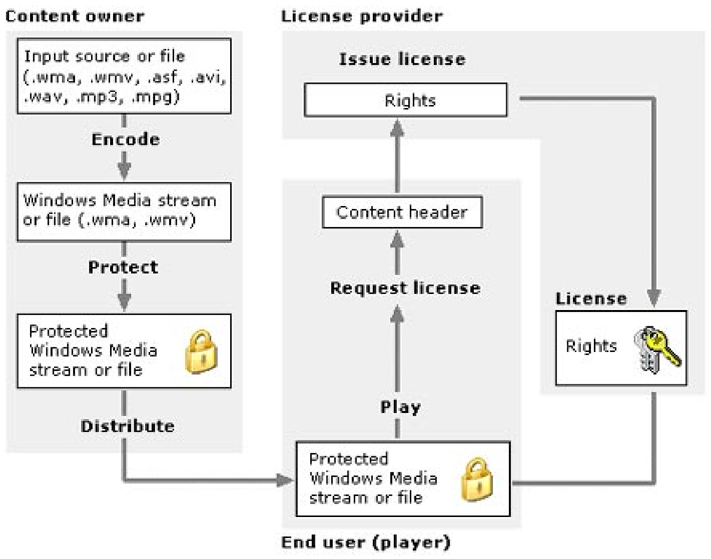

Content owner
License provider
Input source or file (.wma, .wmv, .asf, .avi, .wav, .mp3, .mpg)
Encode
Windows Media stream or file (.wma, .wmv)
Protect
Protected Windows Media stream or file
Distribute
Issue license
Rights
Content header
Request license
License
Rights
Play
Protected Windows Media stream or file
End user (player)

Figure 16. The DRM process. Source [12]

This process is formed basically by 3 steps, which are described next:

Coding/protection and content distribution

The content owner starts by encoding the content (e.g. a song recorded in .mp3 format) into a Windows media stream or file such as .wma or .wmv. The encoding is achieved using Windows Media Encoder 9, where the latter contains also the technology to protect the content, allowing coding and ciphering in one single step.

In order to use Windows Media 9 to protect contents, the content owner must own a DRM profile obtained at a License provider. Examples of license providers can be found at [13].

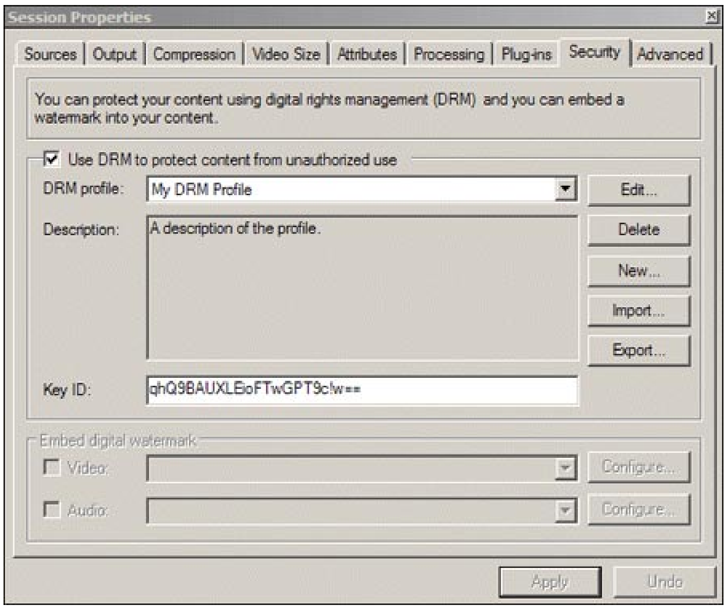
This profile allows the content owner to generate keys in order to protect their media, as well as define the terms of usage. Once protected, the content can be distributed in the usual distribution channels.

During the protection process, Windows Media Encoder generates the key (using the DRM profile), performs ciphering operations and adds DRM specific information to the content. The key is generated using a specific algorithm which uses a license key seed in conjunction with a key id**.**

The license key seed is a value known only by the content owner and by the license authority. Both have to know and share this value in order to protect content. This value is generated when a new DRM profile is being created in a License server.

The “key ID” is a value generated by the content owner and it’s included in the header of the protected file/stream. This is an essential value for the License provider as it’s with this value that the corresponding key is re-generated.

In order to protect content using Windows Media Encoder, the content owner must choose the security tab as shown in the following image:



You can protect your content using digital rights management (DRM) and you can embed a watermark into your content.

[✓] Use DRM to protect content from unauthorized use

DRM profile: My DRM Profile

Description: A description of the profile.

Key ID: qhQ9BAUXLEioFTwGPT9c!w==

[ ] Video:

[ ] Audio:

Figure 17. Protecting content.

Here the user should choose the DRM profile in order to protect the content and a Key ID is automatically generated by Windows media Encoder as one can see in previous figure.

The content owner must also provide the following information on the content header.

- The URL of the license provider
- A unique content ID. It’s a value that can be used to identify the content in a database and related information, such as artist name, date of recording, etc.
- Attributes in the style “name-value” defined by the content owner. E.g. one can include the ciphering date of the content.
- The version of the individualised DRM component. The DRM component is the sub-module responsible for handling all the DRM functions in Windows Media Player. When a user installs a new player, the DRM component is individualised to that user, i.e., this component is unique and machine specific. This way, if a hacker cracked the DRM component would only crack that one, and not all.

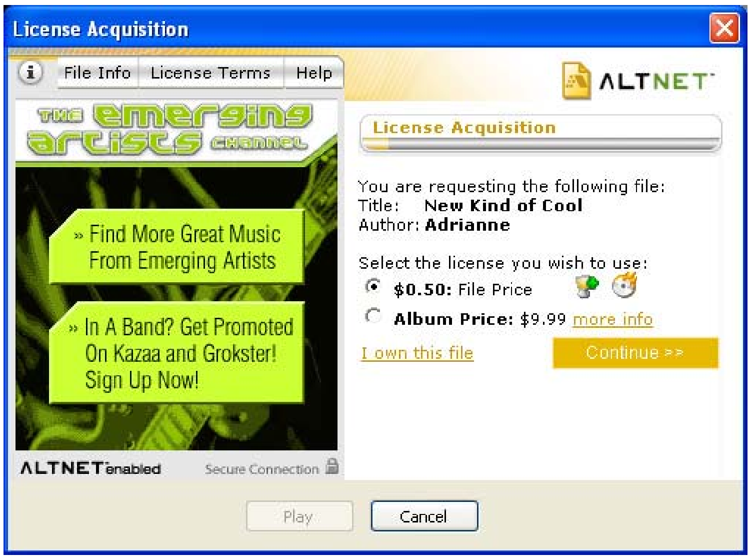
Header signing

As an extra security measure, a public/private key par is used to digitally sign the header of the protected content, in order to ensure that the header and its information are legitimate. For example, a hacker could change the URL of License provider to a site where malicious scripts would be running. In this way, Windows Media player verifies, using the public key, if the header is legitimate or not. If the signature couldn't be verified, the end user would be warned about it. The end-user then chooses whether they want to proceed with the license acquiring process.

Playing protected contents

In order to play protected contents, the end-user must own a player compatible with Microsoft's DRM technology, such as Windows Media Player.

When the end-user tries to play the protected content, Windows Media Player starts by searching in the user's computer for a valid license for that content. If the latter isn't found, the player analyses the content header trying to find the URL of the responsible License Provider. When this is found, the key ID is retrieved from the header and combining it with the license key seed (found at the License provider), the required key to unlock the content is re-generated. A new license is then produced including the key required for playing.



You are requesting the following file:
Title: **New Kind of Cool**
Author: **Adrianne**

Select the license you wish to use:
[✓ **$0.50:** File Price]
Album Price: $9.99

Figure 18. Obtaining a license.

According to the license type, the end-user may perceive the licensing process or not. For example, if a content is for promotional purposes, a license can be issued without any need for registry, being all this operation transparent for the user. This process is called silent registration.

On the other hand, if the content is to be paid, a mini-browser is presented to the end-user showing the licensing conditions and a form for payment information. An example of such mini-browser is presented in the following figure.

Once the License is obtained, the end-user can use the content according to what was defined by the content owner.

License issuing/generation

The process involves two stages:

- Issuing: When a user whished to playback a protected content for which he doesn't own a valid license, a license request is automatically performed to the respective License provider. The location of the license provider can be found at the content's header. The License provider can then identify the content and issue the according license.

- Generation: From the point of view of the content owner, the latter must first choose a License provider and next create a DRM profile. The creation of the DRM profile requires personal information as well as the rights to be included in the licenses to be issued. Once the DRM profile is created, it’s imported to the computer that is responsible for coding and ciphering.

How licenses work

Each license contains the key to unlock the Windows Media file. The license also contains the rights, or rules, that govern the use of the digital media file. The content owner sets these rights to determine which actions are allowed from minimal control over playback to more restrictive licenses. The licenses in Windows Media Rights Manager can support a wide range of different business rules, including:

- How many times a file can be played
- Which devices a file can be played or transferred on. For example, rights can specify if consumers can transfer the file to portable devices that are compliant with the Secure Digital Music Initiative (SDMI).
- When the user can start playing the file and what is the expiration date.
- If the file can be transferred to a CD recorder (burner).
- If the user can back up and restore the license.
- Which security level is required on the client to play the Windows Media file.
- And many others

Licenses can be delivered in different ways and at different times, depending on the business model. The content owner might want licenses pre-delivered, or they might want the license delivered after a consumer has downloaded and attempted to play a packaged file for the first time. Licenses can be delivered with or without the consumer being aware of the process using silent or non-silent license delivery.

Windows Media Rights Manager is a secure technology that helps protect the rights of content owners, while enabling consumers to obtain digital content easily and legitimately.

- Persistent Protection: Windows Media Rights Manager "locks" digital media files with a license key to maintain content protection, even if these files are widely distributed. Each license is uniquely assigned to each computer. This prevents illegal distribution of digital media files.
- Strong Encryption: Windows Media Rights Manager includes proven encryption schemes that ensure distributed digital media files are not exposed to piracy or other illegal use.
- Individualisation: Rights Manager makes each player unique by linking the player to the host computer. This prevents a compromised player from being widely distributed over the Internet. With individualisation, any compromised player can be identified and disabled during the licensing process.
- Secure Audio Path: Rights Manager ensures content protection in the operating system from the player to the sound card driver in Microsoft Windows®

Millennium Edition and Microsoft Windows XP. This secure relationship reduces the likelihood that any unauthorised program will capture a digital media stream within a PC.

- Improved Revocation and Renewability: Windows Media Rights Manager enables the revocation of compromised players when new players become available.
- Secure End-to-End Streaming and Downloads: Digital media files are protected during download and on the consumer's PC through secure cryptographic protocols.

Microsoft Windows Media 10

This version of Windows Media includes new DRM functionality (code named Janus) such as DRM for portable and network devices, supporting synchronisation between any devices that Windows treats as a hard disk (examples of such devices are the Creative Nomad series or RCA Lyra). This is a clear effort to fight Apple’s iTunes music service and its ability to synchronise with iPod devices, as well as to extend the reach of Windows Media to non Windows devices.

The main key features predicted for WMDRM10 are the following:

- WMDRM 10 SDK: Microsoft Windows Media 10 DRM is an SDK entirely written in ANSI C, enabling mobile device producers and developers to write Windows Media DRM dependent applications without having a Windows OS installed in the target machine. According to the capabilities of the target device, the developers use a subset of the Windows Media DRM. However, the OS running on the device is required to support a set of operations required by Windows Media.
- Modes of operation: The WMDRM 10 has two basic modes of operation:
    - Direct License acquisition – Under this mode, the portable device is capable of communicating directly with a license server via the Internet to obtain required licenses for content playback.
    - Indirect License acquisition – When using the mode, the portable device must be docked and connected to a PC in order to obtain licenses.
- License Chaining: The License Chaining structure is a new feature of Windows Media 10, which is used to model and define subscription rights. With this system, users can obtain a “root” license and install it on the device. This license represents basic usage rights for a subscription service, but doesn’t grant any rights to a specific content item. When a user downloads a specific content, a “leaf” license is created and attached to the “root” license, where the latter defines all the usage rights. This system improves substantially interoperability with mobile devices, while satisfying content owners about security.
- Playback counting: This is a controversial feature that WMDRM 10 will support. The system will provide playback counting for a particular content, although users or devices are never identified.
- Secure clock: To use content with time-bound licenses, devices must synchronise a secure clock with a time provider over the Internet or on a computer. This prevents rollback attacks on time-bound content, such as subscription services.
- Automated License Store garbage collector: With this feature, WMDRM 10 provides automatic cleanup of expired licenses, in order to optimise storage space.

2.8 Apple iTunes

iTunes [14] is a music distribution service from Apple directed mainly to Mac users.

In the following sections we will provide some information on the technical features of Apple iTunes Music Store and will be finalised by a short analysis on its key advantages and disadvantages.

Technical Description

In this section we describe the hardware and software required to use iTunes Music Store. We start with the hardware and software requirements. We explain the iTunes4 jukebox software and its dependencies with the iTunes Music Store and the iPod.

Hardware and software requirements

iTunes is available both for Mac and Windows users. The basic requirements to use it follow:

- Mac OS X (version 10.2.5 or later recommended) or Windows 2000, XP or Vista
- Internet connection (DSL, Cable or a LAN-based connection recommended for streaming and downloading music);
- iTunes 4, downloadable from Apple web site.
- QuickTime 6.2

iTunes 4.5 application

iTunes is a music distribution service from Apple directed mainly to Mac and Windows users. In the following sections we will provide some information on the technical features of Apple iTunes Music Store and will be finalised by a short analysis on its key advantages and drawbacks.

Apple iTunes application

Apple iTunes is the application that allows the user to access the iTunes Music Store, an online store with a large selection of music, movies, TV series, podcasts and audio books supplied by the major labels.

iTunes uses the latest AAC audio format and let users share their music with other computers on local Ethernet or AirPort wireless networks (for Mac users).

One should refer that iTunes automatically synchronises with the iPod device (a digital audio player from Apple) at high speeds over FireWire, by connecting iPod to a computer with FireWire. An entire music CD can be downloaded to the device in about 10 seconds.

It is also possible to generate dynamic Smart Playlists that reflect user preferences and listening habits. To create these playlists, a user only has to indicate what kind of music they prefer: iTunes lets user set the parameters — indicating attributes such as My Rating, Genre, Composer, Artist, Play Count, Last Played and so on — and then creates a personalised playlist.

Because iTunes seamlessly connects to the rest of iLife (Apple's software for digital music, photography, movie authoring and DVD creation), it is possible to access iTunes digital music library and playlists from iPhoto, iMovie and iDVD. iTunes can also burn audio CDs, being also fully integrated with Music Store as one can see in the next figure.

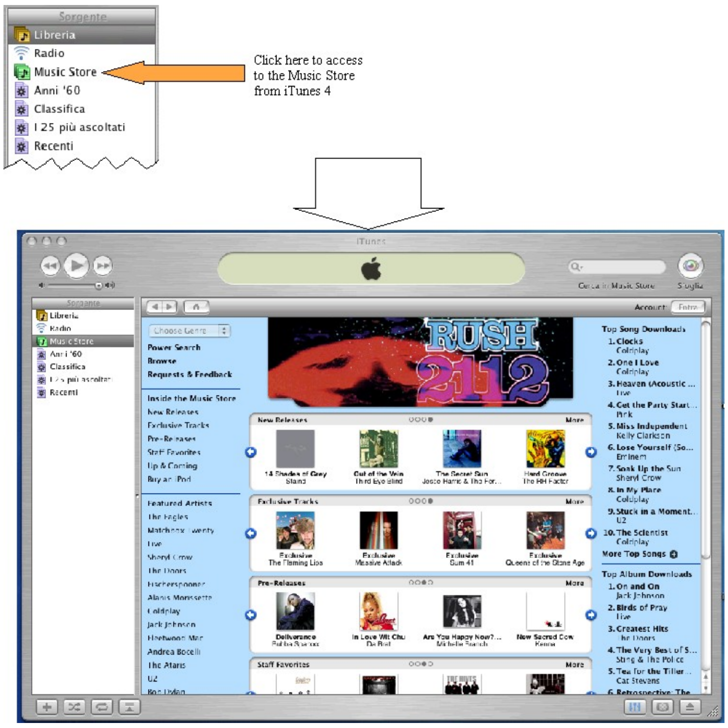


Figure 19. Music Store integration into iTunes.

iTunes Music Store

The iTunes Music Store includes a large number of songs. It is directly accessible via the iTunes software. Due to the direct integration, iTunes Music Store becomes a part of iTunes software which allows users to search or browse genres, new releases, exclusives and more. One should note that any song can be previewed for free, where the price for each song is 0.99$: It can be downloaded immediately on the local hard drive or held in the shopping cart. A user can perform searches by specifying criteria such as artist, composer title and genre. Once satisfied, the user can buy tracks from the returned list, which are downloaded to final users' hard disks. It is possible to purchase both single songs and whole albums, too. The iTunes Music Store also provides artists' discographies and album covers.

For users with a broadband connection, iTunes Music Stores can play full-length music videos. If the end-user doesn’t have a broadband connection, they can download the previews (30 seconds long) to their desktop and listen to them, and then use the shopping cart to hold all their music selection until they are ready to buy. Additionally, there is no limitation in listening to these previews.

The iTunes Music Store also provides the usual email bulletin to keep all the users informed about new releases. To buy tracks in Music Store, a user has to configure his own Apple account from the iTune application itself.

iTunes Accounts and Authorisations

Prior to buying content from the iTunes Store, a user has to create an account with Apple's servers and then authorise a PC or Mac running iTunes.

During authorisation, iTunes creates a globally unique ID number for the computer it is running on, then sends it to Apple's servers, where it is assigned to the user's iTunes account. Five different machines can be authorised.

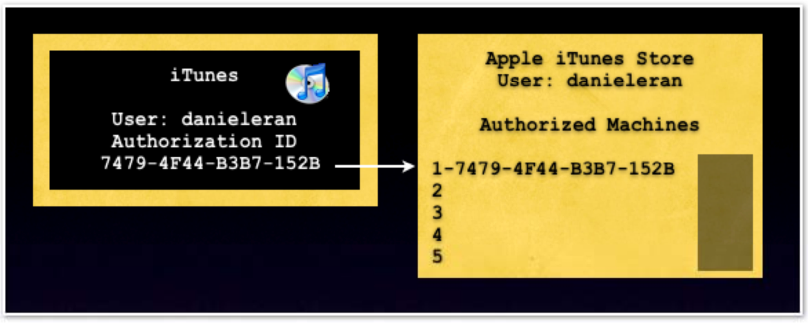


Figure 20. Authorised devices in iTunes.

When a user buys a song from the iTunes Store, a user key is created for the purchased file. The AAC song itself is scrambled using a separate master key, which is then included into the protected AAC song file. The master key is locked using the user key, which is both held by iTunes and also sent to Apple’s servers.

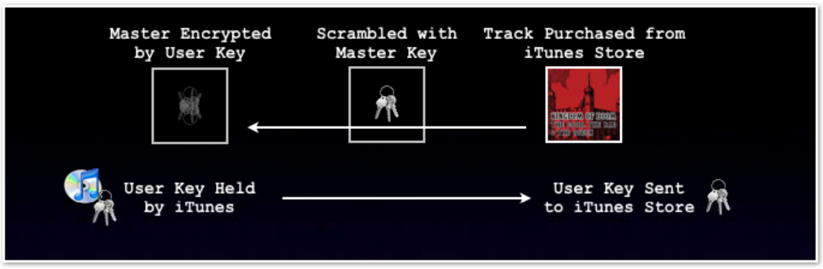

Master Encrypted by User Key
Scrambled with Master Key
Track Purchased from iTunes Store
User Key Held by iTunes
User Key Sent to iTunes Store

Figure 21. iTunes user resource encryption with user key.

Protected, purchased content is locked within iTunes; songs are not scrambled on Apple's server. This speeds and simplifies the transaction by delegating that work to iTunes on the local computer.

The result is an authorisation system that does not require iTunes to verify each song with Apple as it plays. Instead, iTunes maintains a collection of user keys for all the purchased tracks in its library.

To play a protected AAC song, iTunes uses the matching user key to unlock the master key stored within the song file, when is then used to unscramble the song data.

Every time a new track is purchased, a new user key may be created; those keys are all encrypted and stored on the authorised iTunes computer, as well as being copied to Apple's servers.

When a new computer is authorised, it also generates a globally unique ID number for itself and sends it to Apple, which stores it as one of the five authorisations in the user account.

Apple's server sends the newly authorised machine the entire set of user keys for all the tracks purchased under the account, so all authorised systems will be able to play all purchased songs.

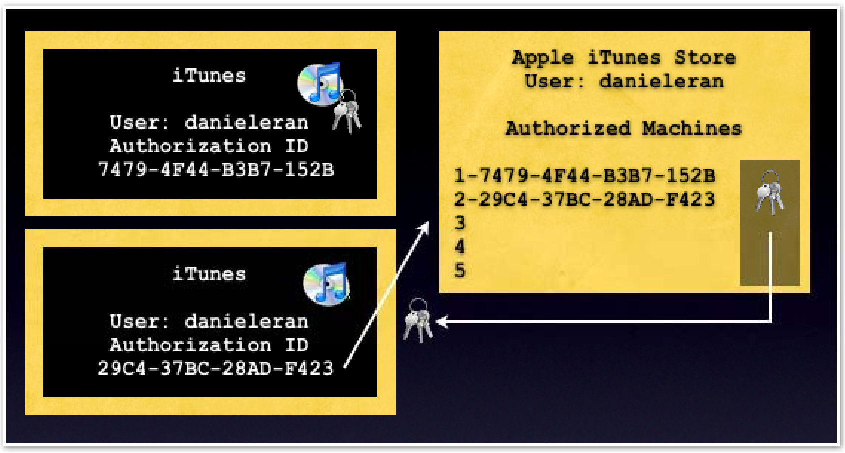


Figure 22. Adding a new authorised machine in iTunes.

An iTunes computer can be authorised by multiple iTunes user accounts; for each account, iTunes maintains a set of user keys.

Exploiting Authorisations in FairPlay

When a computer is deauthorised, it deletes its local set of user keys and requests Apple to remove the authorisation from its records.

If the keys are backed up, users can deauthorise their systems, then restore the keys and authorise a new set of computers, resulting in more than five machines that can all play the existing purchased music.

However, any new music purchased on the newly authorised systems will create new keys, and the previously de-authorised machines will not be able to play the new purchases because they can't obtain the new keys.

iTunes Keys on the iPod

Any number of iPods can be used with an authorised computer running iTunes. Once an iPod is connected, it downloads all the user keys from iTunes so it can unlock and play any protected tracks. If that copy of iTunes is authorised to play songs from multiple accounts, all of the accounts' user keys are uploaded.

The iPod makes no decisions about which tracks it can play, it simply is given user keys for all the songs it contains by iTunes.

If iTunes has songs in its library, but lacks the keys to play them (from another account or on a deauthorised computer that has dumped its keys) it will simply not copy the protected songs to the iPod.

There is no way unplayable protected songs can be copied to the iPod without the user keys to play them, because iTunes will not let this happen. This again delegates the burden of DRM to iTunes, making the iPod simpler.

That also explains why users can't dock a single iPod with different users' iTunes and suck up all their music; the only option available is to replace the music on the iPod with the music from the new iTunes library.

Since iTunes manages all the music on an iPod, there is no way to synchronise an iPod with multiple iTunes libraries; the iPod simply wasn't given the intelligence to mange multiple libraries.

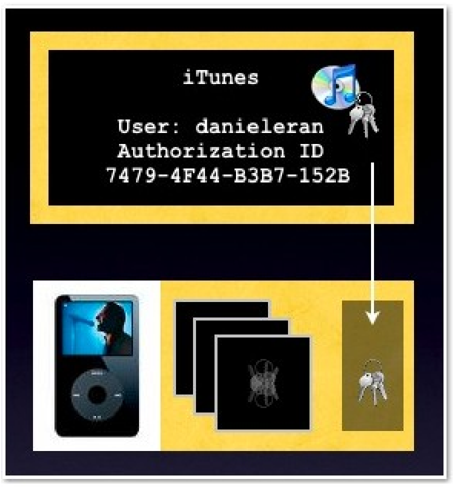


Figure 23. Copying protected content to the iPod in iTunes.

With iTunes 7 however, Apple added the ability for an iPod registered with an iTunes account to synchronise purchased songs with any of the five machines authorised by that account. Each copy of iTunes can update the user keys on the iPod and add new purchased tracks, ensuring that the iPod can play all the music copied to it.

Analysis of iTunes

In this section we will analyse some of the key advantages of iTunes as well as some of its drawbacks.

The main advantages are:

- DRM restrictions are weakened compared to other distribution systems. This is a key to success as the technical restrictions must never annoy the end-user. Apple considered this requirement while designing the iTunes Music Store.
- iTunes is easy to use.
- Besides the usability, the costs are relevant to users: inexpensive music prices are advantages for the success and 99 cents for each is not expensive
- There are no costs for subscription.
- Content providers are the most important music labels nowadays: The major labels (Universal, Warner, BMG, EMI and Sony Music Entertainment) provide Apple iTunes Music Store with music.
- First system to gather selling and downloading songs, burning them onto CDs and transferring to portable music players.
- Songs are not disabled when subscription ends.

The main disadvantages are:

- If a user is not interested in one or two individual files of an album but prefers to download almost the complete album buying a CD will be more attractive because of the lack of bulk discounts in the iTunes Music Store, the added value of a CD like the backup and the cover art, the unrestricted CD and maybe due to some bargains at local record shops.

- The quality of encoding is restricted to 128kbit AAC encoding which is comparable with 160 kbit MP3 encoding. However, some analyses show that 256 kbit MP3 encoding is necessary to achieve full CD quality.
- There is a noticeable loss of quality when converting the AAC files to MP3 files. This also endangers the exchange with other MP3 players.
- An advanced search engine might be more interesting (with more than one search field).
- The iTunes Music Store is only available in the U.S. and Europe.
- Apple commands less than 5 percent of the desktop computer market.

2.9 CORAL

The objective of Coral [15] is to facilitate the creation of DRM interoperability services, by third parties who have incentive to do so, which work among existing DRM technologies.

The specifications consist on the Coral Core Architecture for DRM interoperability and a specification called Ecosystem-A which extends the Core Architecture for home networking interoperability.

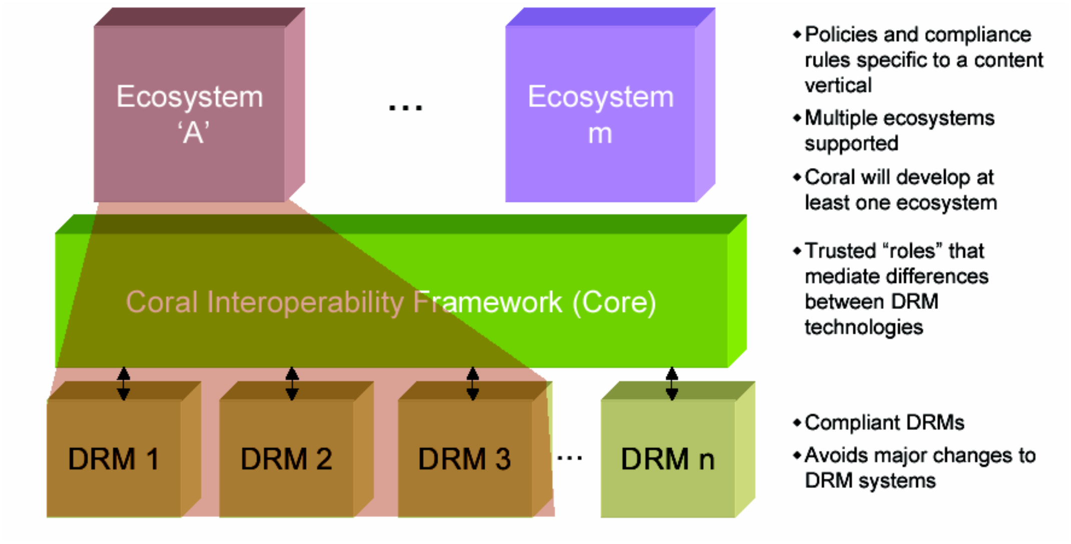

Ecosystem 'A'
...
Ecosystem m
Coral Interoperability Framework (Core)
DRM 1
DRM 2
DRM 3
...
DRM n
Policies and compliance rules specific to a content vertical
Multiple ecosystems supported
Coral will develop at least one ecosystem
Trusted "roles" that mediate differences between DRM technologies
Compliant DRMs
Avoids major changes to DRM systems

Figure 24. Coral Interoperability Core and Ecosystems. Source: [16]

Coral itself is not a DRM system, but provides the means to interoperate with different DRM systems.

The Coral Core Architecture provides an underlying infrastructure that must be completed by the design of a content ecosystem responsive to the needs of a particular deployment. To look at an informative analogy, the Coral Core Architecture provides a measure of interoperability for DRM systems in the same way that the HTTP protocol and HTML language provide interoperability for the World Wide Web. These protocols define behaviours that must be implemented by all web browsers and web servers, but do not constrain the uses to which these technologies are applied. Similarly, designers of ecosystems based on the Coral Core Architecture must define the specific application of the Core technologies.

The elements that must be defined by a Coral ecosystem include:

- Usage Models: A common set of usage models that must be enforced by all content protection systems that are allowed to participate in the ecosystem. For example, if the ecosystem is designed to support time-based subscriptions, support for the secure management of time is required. This specification defines a domain model that requires specific domain semantics that are not specified in the Coral Core Architecture.
- Policies: An ecosystem may define certain parameters that modulate the behaviour of the system for security or usability reasons. An ecosystem might, for example, support the notion of periodic synchronisation with a certificate revocation list to ensure that illegitimate actors are quarantined. The frequency of such synchronisation might be a policy variable.
- Roles: The Coral Core Architecture defines a Role as an assertion that a particular entity engages in a standardised set of behaviours defined by the Coral Core Architectures or by ecosystems built upon it. Possessing a particular Role provides the predictability needed for interoperability across implementors. If a particular entity can be verified to possess a given role, then the other systems with which it interacts can be certain that it will behave according to a specification. The Coral Core Architecture defines some Roles; ecosystems may define their own Roles as necessary to provide ecosystem-specific semantics or functionality.
- Role Groupings: An ecosystem may additionally specify dependencies amongst Roles, both for Roles that it defines itself and Roles that are incorporated from the Coral Core Architecture. To take an example from this specification, an entity with the Ecosystem-A Domain Manager Role must also possess the Principal Relation Manager role defined in the Core Architecture. Role grouping allows ecosystem designers to define their systems by extending and composing existing roles, adding ecosystem-specific functionality where required.
- Extensions: The Coral Core Architecture is designed to be extensible in as many ways as possible, allowing ecosystem designers to enhance the underlying framework for a specific application. In Ecosystem-A specification, for example, opaque DRM-specific information is conveyed in the extension fields of interfaces defined in the Coral Core Architecture.
- Third-party Technologies: Ecosystems may incorporate or require the use of third-party technologies not present in the Coral Core Architecture. For example, specific Secure Authenticated Channel or other output technologies may be incorporated into the ecosystem specification.

Coral's core architecture includes roles, which are functions, and nodes, which are physical realisations of collections of roles.

Two important roles in the Coral architecture are the Content Mediator and Rights Mediator roles, which determine whether one system should be allowed to exercise rights on a particular piece of content from another system. Other roles include DRM Content Exporter, DRM Content Importer, and DRM License Mapper. The latter maps rights in one system to rights in another by making reference to a set of policies.

Nodes are combined into Ecosystems, which are collections of nodes that are considered mutually trustworthy so that they can interface with one another without security concerns that lead to diminution of functionality. The entire architecture is based on NEMO (Network Environment for Media Orchestration), a secure messaging scheme based on X.509 identity certificate and SAML security assertion standards.

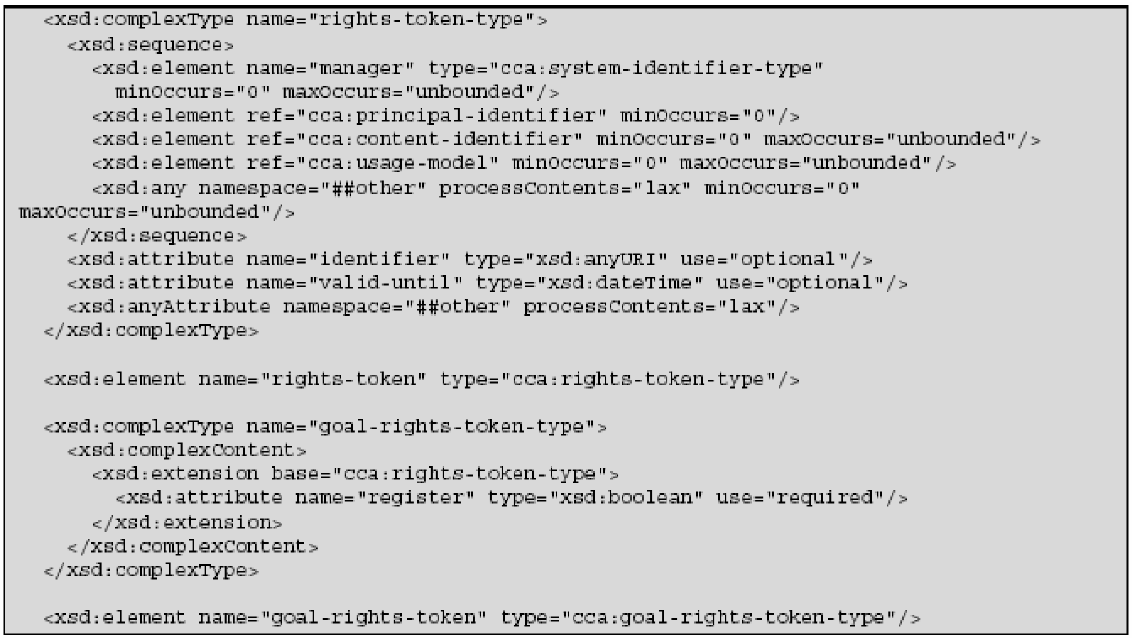
Ecosystem-A instantiates and extends the Coral core architecture for the specific application of content interoperability in home networks. It implements, among other things, two important concepts:

- rights lockers: repositories of licenses associated to content that principals (users and/or devices) hold
- domains: groups of devices that act as single principals for licensing purposes

In order for Coral to interoperate between different DRM systems it is necessary that those vendors modify their technologies to make them Coral-compliant.

When a user buys a certain content, Coral system registers a right token in a rights locker designed by the user. When requested, the rights locker sends the rights token to the license service with which a device interacts in order to check the rights fulfilment. If the same content needs to be consumed on a different device that works with a different license service, then rights mediator will request the rights locker the token and send it to the new license service, which will use it to construct the corresponding native DRM license and send it to the new device.

Coral approach for solving DRM license interoperability in order to obtain equivalent licenses for different systems is to derive them from a common rights token. This approach supposes the definition of a new rights expression language from which other languages can be derived. In fact, Coral provides in its specifications the XML Data Type Definition (DTD) for the creation of such rights tokens.



```
  <xsd:complexType name="rights-token-type">
    <xsd:sequence>
      <xsd:element name="manager" type="cca:system-identifier-type"
        minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="cca:principal-identifier" minOccurs="0"/>
      <xsd:element ref="cca:content-identifier" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="cca:usage-model" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="identifier" type="xsd:anyURI" use="optional"/>
    <xsd:attribute name="valid-until" type="xsd:dateTime" use="optional"/>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>

  <xsd:element name="rights-token" type="cca:rights-token-type"/>

  <xsd:complexType name="goal-rights-token-type">
    <xsd:complexContent>
      <xsd:extension base="cca:rights-token-type">
        <xsd:attribute name="register" type="xsd:boolean" use="required"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:element name="goal-rights-token" type="cca:goal-rights-token-type"/>
```

Figure 25. CORAL Rights token DTD. Source: [17]

2.10 DMP

The Digital Media Project (DMP) [18] approaches the problem of DRM Interoperability by specifying technologies (called Tools) required to implement “Primitive Functions”, i.e. “smaller” Functions obtained by breaking down the Functions that Value-Chain

Users perform when they do business between themselves. It is expected that, while Functions may undergo substantial changes as a consequence of the evolution of the media business, Primitive Functions will generally remain stable.

Therefore, far from targeting a universal "DRM standard" capable of providing Interoperability between Users in arbitrary Value-Chains or across different Value-Chains, DMP only provides:

- Specifications of Tools enabling Primitive Functions
- Examples of how Value-Chains serving specific goals can be set up using the standard Tools.

The Digital Media Project (DMP) has developed three versions of the Interoperable DRM Platform specification: IDP-1, IDP-2 and IDP-3 [47]. Chillout [48] is the name of the IDP Reference Software, released as Open Source Software under Mozilla Public Licence 1.1, which is under development.

The DMP Architecture consists of the following parts:

Value-Chain

A typical Value-Chain has the following main components:

- Creation (including Adaptation)
- Instantiation
- Production
- Content Provisioning
- Service Provisioning
- Consumption

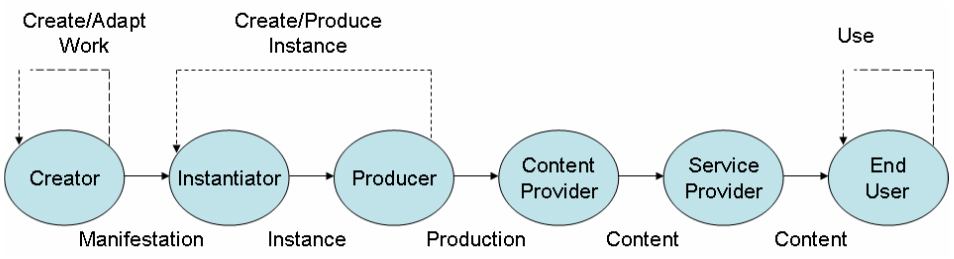


Figure 26. General schematic case of a Value-Chain. Source: [47]

Walkthrough

For proper management in a Value-Chain it is useful to combine different types of Resources with different types of Metadata and possibly other information types. DMP calls this combination Content. The digital Representation of Content, needed to Use Content on Devices, is called DMP Content Information (DCI).

A User wishing to express conditions to Use a Content Item can associate a Licence to it Granting Permissions under specified Conditions. In this case the Content is called Governed Content. The party Granting Permissions is referred to as Licensor and the party receiving them is referred to as Licensee. A User who does not wish to express Conditions to Use a Content Item can do so by Releasing it without a Licence.

To enable a Device to interpret Permissions without human intervention, a machine readable language called Rights Expressions Language (REL) is needed. The Licensee can be a Device, a User or a Domain and the Licence can be Bundled within the Content (i.e. it is part of the DCI) or not Bundled within the Content (i.e. the DCI references an external License). Other Content Elements (e.g. Resources) can be in-line or referenced.

The Content Elements in Governed Content can either be in a form that allows immediate Processing, e.g. for Rendering by a Device (so-called Clear-text) or in a protected (i.e. Encrypted) form. Keys and related DRM information can be conveyed by various means.

When the Device has limited capabilities (as in PAVs) it is useful to be able to make reference to a basic selection of Encryption Tools. However, when the Device does not have such restrictions (as in SAVs) it is important to be able to convey blocks of executable code (called DRM Tools) in a DCI that are required to Process various types of Governed Content.

XML is the technology selected by DMP to Represent Content. XML is very powerful and flexible but Content Represented by means of XML can easily become bulky and unwieldy. Therefore DMP has selected an XML binarisation technology that not only reduces the size of a DCI but also allows simpler Processing in a Device without loss of information.

To Deliver Content between Device Entities it is necessary to Package a DCI (in binary form) and its referenced Resources. Two forms of Delivery are possible: as File (DMP Content File - DCF) and as Stream (DMP Content Broadcast - DCB - in the case of an MPEG-2 Transport Stream, and DMP Content Streaming - DCS - in the case of Real Time Protocol on Internet Protocol).

In general Users require Devices to perform the Functions proper of their role in the Value-Chain. To entrust their Content to a Device, Rights Holders must have assurance that the Device will execute Functions as expected. Device Certification is the process that provides that assurance. This is performed by a number of organisations (Certification Agencies) that are properly appointed and overseen by a single root authority (Certification Authority). DMP appoints the Certification Authority after approving the Authority's Certification policies. The figure below depicts this three-layer arrangement.

In general interacting Devices require the establishment of a Trust relationship between them, e.g. when they Deliver Content between them. A precondition for this to be possible is that a Device be Identified. The task of Assigning Identifiers to Devices is performed by a number of organisations (Registration Agencies) that are appointed and overseen by a single root authority (Registration Authority). DMP appoints the Registration Authority after approving the Authority's Registration policies. The same three-layer arrangement used for Certification is also used for Identification.

Content Items also require Identification. When Identifiers are Assigned appropriate Metadata are also typically generated.

DMP Models

DMP's IDP-2 provides the following models:

- Creation Model: identifying the major entities for which IP is attributed and describing their relationship to the digital objects involved in Content Creation. The following objects referred to as IP Entities are defined formally in the DMP Terminology: Work, Adaptation, Manifestation, Instance, Expression.

- Distribution Model: identifying and describing the role of the principal Value-Chain Users engaged in distribution. In the general DMP Distribution Model the following Users operate: Content Providers, License Providers, DRM Tool Providers and Service Providers.
- Delivery Model: describing the two basic ways (File and Stream) in which Content can be Delivered between Devices. DMP defines Delivery as a File (DCF), on an MPEG-2 Transport Stream (DCB) and on a Real-Time Protocol transport (DCS).
- DRM Tool Model: describing the general operation of DRM Tools in Devices. DRM Tools required to e.g. decrypt Resources may be natively embedded in a Device and executed when such DRM Tools are required. Alternatively DRM Tools may be Delivered as part of a DCI. If the DCI does not contain the required DRM Tool, the DRM Processor in the device tries to Access it from the local Secure Storage. If the required DRM Tool is not found there, the DRM Processor Accesses the missing DRM Tool from the DRM Tool (or Service) Provider. In addition to individual DRM Tools, IDP-2 gives the possibility to convey DRM Tool Packs in a DCI. A DRM Tool Pack is composed of a DRM Tools Agent and a set of DRM Tools called DRM Tool Group. A DRM Tool Pack may contain all the DRM Tools required or require other DRM Tools external to the Tool Pack.
- Domain Model: describing the operation of Domains of Devices and Users. Using Domains it becomes possible to implement more flexible Licensing modalities, e.g. to License a Content Item to all Devices or Users in a Domain.
- Device Model: identifying and describing the principal Devices employed by Value-Chain Users, e.g. Content Consumption Device (PAV), Content Consumption Device (SAV), Content Creation Device, Content Identification Device, Content Provider Device, Device Identification Device, DRM Tool Identification Device, Domain Identification Device, Domain Management Device, DRM Tool Provider Device, License Identification Device, Lic. Provider Device, PAV eXternal Device.
- Import/Export Model: describing how governed Content can be converted to Governed Content from a different value chain and vice versa, by using mechanisms such as translating licenses.
- Data Model: identifying and describing the different types of Data specified by DMP, as for example, Content, Identifier, Resource, Metadata, DRM Information, DRM Tool, DRM Tool Body, Licence, Key, DRM Message, Device Information, Domain Information, Use Data, DCI Signature, DCI Hash.

Part III. Contribution

3 Contribution

3.1 Contribution summary

The main goal of this research project is to participate in the definition of a generic architecture for the protection and management of digital rights in a multimedia content management scenario following open standards as much as possible. This ambitious and broad aim is included in a general research framework, which is being carried out by the Distributed Multimedia Applications Group (DMAG).

In order to achieve our goal, several standards and solutions need to be analysed and taken into account. A relevant standard to be considered is MPEG-21, as it specifies a multimedia framework that provides interoperability among systems that deliver multimedia content. On the other hand, the requirements of other *de facto* standards and initiatives will also be analysed so as to define a general architecture that is not only restricted to manage a specific kind of content or format.

The DMAG long experience in the MPEG-21 area has been exploited in this sense to develop several new software tools that integrate additional functionalities together with the already existing DMAG software modules in order to show the results and implications of the integration work. The already existing software tools correspond to different research works performed under the same research group framework.

The first contribution involves the integration two of the MPEG-21 standard parts: Digital Item Declaration (DID), Rights Expression Language (REL) and Rights Data Dictionary (RDD). The work has involved the identification of how rights expressions can be embedded and extracted from the digital objects to which they apply, and how the linkage between the resource referred from the license and that included in the digital object is achieved. In this sense, two software contributions that integrate the new functionality together with the license-based authorisation mechanisms will be presented.

The second contribution in this sense goes one step further, and integrates the previous DID and REL work together with the Digital Item Processing (DIP) part. The DIP part enables the management of multimedia information related to many of the MPEG-21 parts. In this context, we have analysed the relationship of DIP and the implications of its development mainly related to the DID, REL and RDD parts. In this sense, we have developed a specific software by which the advantages and disadvantages of implementing the REL-related functionality at different levels of the DIP specification have been evaluated. This software has been developed as MPEG-21 core experiment, which is the mechanism provided by MPEG-21 to investigate if new functionalities need to be added to the standard, and which are the best solutions to adopt. After the core experiment results discussion, we have proposed the standardisation of the interfaces of two new elements in the DIP part to support some functionalities related to license retrieval and authorisation of users integrated in the DIP context. These two interfaces have been accepted and added to the DIP specification.

Once complete these integration tasks in the MPEG-21 standard, the definition of a preliminary architecture for the protection and management of digital rights for multimedia content has been tackled. This preliminary architecture was mainly based on the MPEG-21 standard, as a result of the integration work that had been previously performed.

Later on, we have realised that the architecture definition should not be restricted only to MPEG-21 data, but be general and flexible enough to support the requirements of different standards. In this sense, the requirements of other *de facto* standards and initiatives have been considered so as to define a general architecture that is not only restricted to manage a specific kind of information. To complete this task we have considered the requirements of relevant standards, initiatives and projects such as the Open Mobile Alliance (OMA), Open Mobile Alliance Digital Rights Management (OMA DRM), MPEG-21, Digital Media Project (DMP), Networked Audio Visual Systems and Home Platforms Digital Rights Management (FP6 NAVSHP DRM) and Automating Production of Cross Media Content for Multi-channel Distribution (AXMEDIS) European project, which have helped to identify the necessary modules and their functionality in the new architecture.

The specification and implementation of the generic architecture, called MIPAMS, has been done in the context of several Spanish and European projects such as AXMEDIS, VISNET II, Linked-Work and GILDDA. After the specification of the MIPAMS architecture, which includes the description of the integral modules, use cases and mappings and comparisons to other initiatives, in subsequent sections we give the details and peculiarities of each of the projects where it has been used or will be used in the near future.

Whereas in the AXMEDIS we focus on the security and testing activities, in VISNET II and GILDDA we present the use cases where the architecture will be deployed. Finally, in Linked-Work section, we detail the specific implementation that has been tackled, emphasising on the key concepts that make it different from other kind of DRM systems. In this latter section we provide a thorough analysis of the Linked-Work specification, implementation and use cases.

As a result of the usage of standards in the developed architectures, several issues have arisen regarding their applicability. Chapter 3 describes the issues regarding the usage of MPEG-21 Event Reporting, which relate to the distributed generation of Event Reports. Our contribution has been considered, accepted and included in the resulting corrigendum [49], having a direct impact on the text of the standard.

4 Integration of different parts of MPEG-21

4.1 Motivation

The work on the integration of the MPEG-21 parts is a very important and necessary step to develop MPEG-21 applications that show the interrelation between different parts of the standard. The first aim of this work is to widen the contribution made by the DMAG group in the MPEG-21 standardisation process by covering new parts, and their interrelation. On the other hand, another very important aim of this work is to obtain new results from the contributions that provide the necessary knowledge and expertise related to MPEG-21 that helps to go one step further and work in the definition of an architecture for the management and protection of multimedia information. The results of the latter goal will be presented in subsequent chapters.

4.2 Integration of DID, REL and RDD

Based on the previous experience of the DMAG group on rights expression languages (RELs), and mainly on MPEG-21 REL, the first step has consisted on the integration of DID, REL and RDD. Section 4.2.1 presents the first integration of DID and REL, which involves license embedding and retrieval from/to DIDL documents together with simple authorisation mechanisms. Section 4.2.2 presents the enhanced version of the first integration, which solves some aspects that were not considered in the first approach and includes the RDD facilities. Both software packages were contributed to the MPEG-21 meetings as input documents [50] [51] and have been included in the MPEG-21 reference software part [26] as informative modules, which are aimed to show the MPEG-21 standard usage.

4.2.1 First version of the DID and REL integration

The DID-REL integration has involved the analysis of the placement of licenses within DIDL documents, the mechanisms through which they can be extracted from them, and the integration of the mentioned aspects together with authorisation mechanisms.

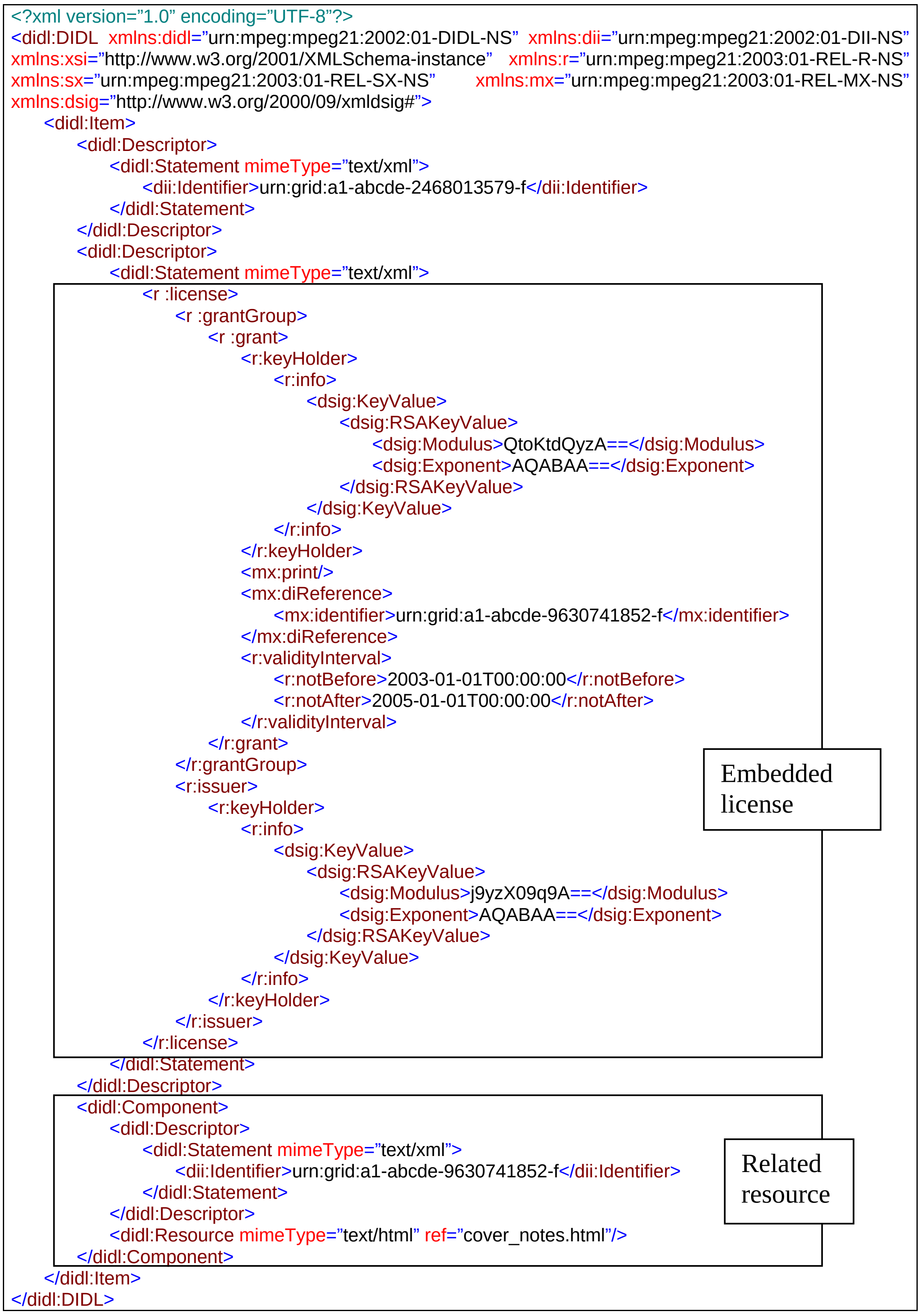
The DID-REL integration has been materialised in the "MPEG-21 REL license interpretation within DID" [50] contribution to MPEG-21, which has been implemented as described in the MPEG-21 REL/RDD Software Implementation Plan [52], where it is included. It consists in a software module that, given a DID expression with any REL licenses embedded and a query consisting of principal A, right B, resource C, time D and exercise count E, answers the question "is the user A authorised to exercise the right B over the resource C under the conditions D and E?".

4.2.1.1 License placement within the DIDL document

At the moment when this first integration of DID and REL was performed, the only available example of how to integrate a license embedded n a DIDL document was included in the Annex I of the REL FDIS part of MPEG-21. This was a merely informative annex, but it was taken as a model to develop the integration of DID and REL. Later on, the Microsoft's version for the contribution to the integration of DID and REL used the same structure to place the licenses in the DIDL documents.

The way to embed a license in DIDL documents takes profit of the Statement element. A Statement element can include any kind of information, thus providing the way to embed licenses. The Statement element where the license is placed must be a child of a

Desriptor element that refers to a Component element in which the resource to which the license refers is included.



```
<?xml version="1.0" encoding="UTF-8"?>
<didl:DIDL  xmlns:didl="urn:mpeg:mpeg21:2002:01-DIDL-NS"  xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"   xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
xmlns:sx="urn:mpeg:mpeg21:2003:01-REL-SX-NS"       xmlns:mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
    <didl:Item>
        <didl:Descriptor>
            <didl:Statement mimeType="text/xml">
                <dii:Identifier>urn:grid:a1-abcde-2468013579-f</dii:Identifier>
            </didl:Statement>
        </didl:Descriptor>
        <didl:Descriptor>
            <didl:Statement mimeType="text/xml">
                <r :license>
                    <r :grantGroup>
                        <r :grant>
                            <r:keyHolder>
                                <r:info>
                                    <dsig:KeyValue>
                                        <dsig:RSAKeyValue>
                                            <dsig:Modulus>QtoKtdQyzA==</dsig:Modulus>
                                            <dsig:Exponent>AQABAA==</dsig:Exponent>
                                        </dsig:RSAKeyValue>
                                    </dsig:KeyValue>
                                </r:info>
                            </r:keyHolder>
                            <mx:print/>
                            <mx:diReference>
                                <mx:identifier>urn:grid:a1-abcde-9630741852-f</mx:identifier>
                            </mx:diReference>
                            <r:validityInterval>
                                <r:notBefore>2003-01-01T00:00:00</r:notBefore>
                                <r:notAfter>2005-01-01T00:00:00</r:notAfter>
                            </r:validityInterval>
                        </r:grant>
                    </r:grantGroup>
                    <r:issuer>
                        <r:keyHolder>
                            <r:info>
                                <dsig:KeyValue>
                                    <dsig:RSAKeyValue>
                                        <dsig:Modulus>j9yzX09q9A==</dsig:Modulus>
                                        <dsig:Exponent>AQABAA==</dsig:Exponent>
                                    </dsig:RSAKeyValue>
                                </dsig:KeyValue>
                            </r:info>
                        </r:keyHolder>
                    </r:issuer>
                </r:license>
            </didl:Statement>
        </didl:Descriptor>
        <didl:Component>
            <didl:Descriptor>
                <didl:Statement mimeType="text/xml">
                    <dii:Identifier>urn:grid:a1-abcde-9630741852-f</dii:Identifier>
                </didl:Statement>
            </didl:Descriptor>
            <didl:Resource mimeType="text/html" ref="cover_notes.html"/>
        </didl:Component>
    </didl:Item>
</didl:DIDL>
```

Figure 27. How a license can be embedded in a DIDL document.

Finally, with the development of the IPMP part of MPEG-21, a normative element for the license placement has been defined, enabling more advanced functionalities, as the license protection, amongst others.

4.2.1.2 Architecture

The software implementation consists of two main modules: a License extractor, and the previously implemented DMAG REL License Interpreter [53]. Kawa's implementation [54] of XQuery [55] has been used to extract the resource to check from user documents and the matching licenses from the DID. Next, we describe the License extractor, which is the new module developed in the context of this research work.

First of all, the resource (in a url or urn format) is extracted from the User Query file. Then, the DID document is traversed to identify the resource. If the resource in the Query is in url format (e.g. cover_notes.html), a urn identifier is sought. If the resource in the query is in urn identifier format (e.g. urn:grid:a1-abcde-9630741852-f), a url that matches the same resource is sought. Finally, the DID document is traversed to find all the licenses that match the resource (in both formats: url and urn). If a matching `license` or `licenseGroup` is obtained, then a temporal LicenseFile file is created, the REL License Interpreter is invoked and an Output file will be given as the output. Otherwise, the License Extractor directly gives the Output file. We have assumed that the REL `schemaLocation` can be defined either in the `DIDL`, `license` or `licenseGroup` tag.

Figure 28 shows the structure of our solution, that we have named "DMAG REL License Interpretation within DID" after our group acronym (DMAG).

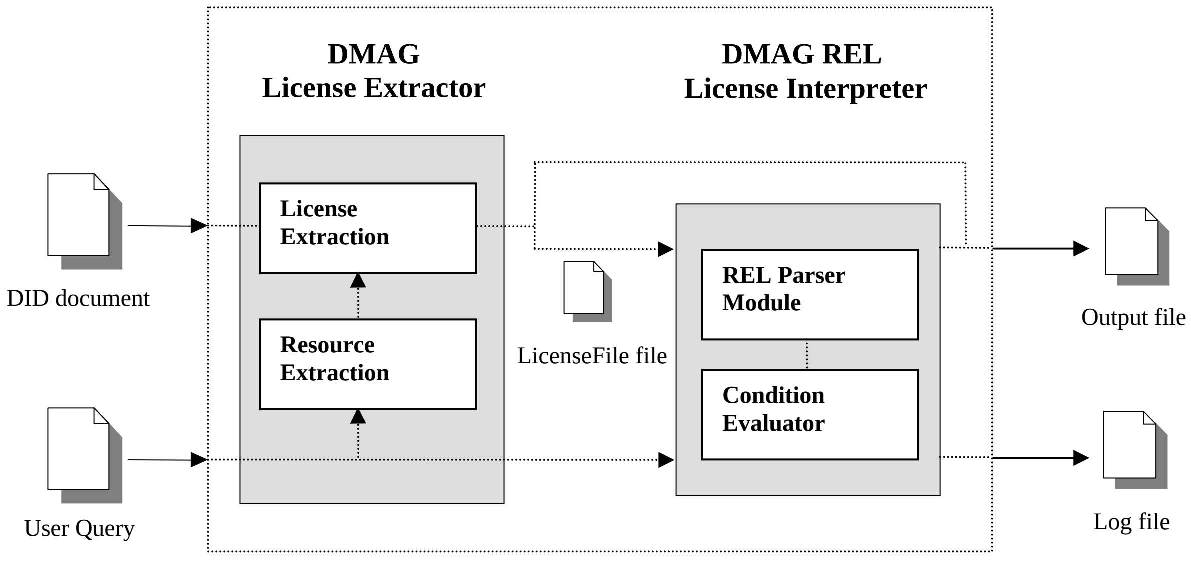


Figure 28. DMAG REL license interpretation within DID.

4.2.1.3 Structure of the software

As we said before, the software implementation that we propose consists of two main modules:

License Extractor

It obtains the resource from the user's query and looks for matching grants in the DID.

- Inputs:
    - A well formed DID document. Every `license` or `licenseGroup` must be a `Statement` child. For the moment, all grants must have a `diReference` or a `nonSecureIndirect` element to identify the resource they match to.
    - A User Query. For the moment, the resource must be in urn or url format.
- Outputs:
    - If matching grants are found, the output is a LicenseFile file which contains the license(s) with the grants that match the resource.
    - If no matching grants are found, the output is an Output file saying that no matching grants were found.

REL License Interpreter

Refer to [53] for further information on the DMAG REL License Interpreter architecture and operation. Another researcher in the DMAG research group has developed it.

- Inputs:
    - A License.
    - A User Query.
- Outputs:
    - If there are conditions associated to the matching grants, the output is an output file saying whether the conditions are satisfied or not.
    - If there are no conditions associated to the matching grants, the output is an output file saying there are no conditions associated to the matching grants.
    - A Log file if any problem occurs.

4.2.1.4 Software Details

- Languages: Java, XML Query.
- Parser (only for the License Interpreter): Xerces DOM.
- Supported Platforms: Windows, Linux.

4.2.1.5 Available tests

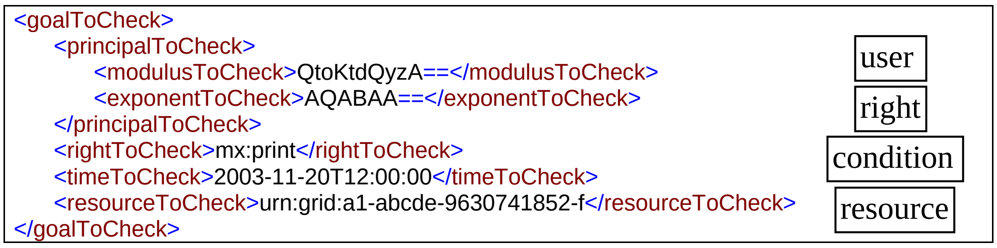
Together with the software several sample test files have been provided. They illustrate the software operation using different DID document and User Query files.

Table 6 shows the main features of each test. In the User Query file the resource is expressed in urn or url format. In the DID document, although all sample tests include the REL Namespaces and REL Schema Location definitions in the same `DIDL`, `license` or `licenseGroup` tag, the designed software would accept them to be defined separately. In tests 1 to 6, any `licenseGroup` element is present whereas in tests 9 and 10 there is a single `licenseGroup`. Different scenarios have been proposed, where the Principal, Resource, Right and Condition to check from the User Query may or may not match those in the DID document.

Table 6. Sample test main characteristics.

	User Query	DID document		Matching DID document – User Query			
Test	Resource format	REL Namespaces and REL Schema Location definitions	LicenseGroup element	Principal	Resource	Right	Condition
1	urn	`DIDL` tag	NO	YES	YES	YES	YES
2	url	`license` tag	NO	YES	YES	YES	YES
3	urn	`DIDL` tag	NO	NO	YES	YES	YES
4	urn	`DIDL` tag	NO	YES	NO	YES	YES
5	url	`license` tag	NO	YES	YES	NO	YES
6	url	`license` tag	NO	YES	YES	YES	NO
7	urn	`DIDL` tag	YES	YES	YES	YES	YES
8	url	`licenseGroup` tag	YES	YES	YES	YES	NO

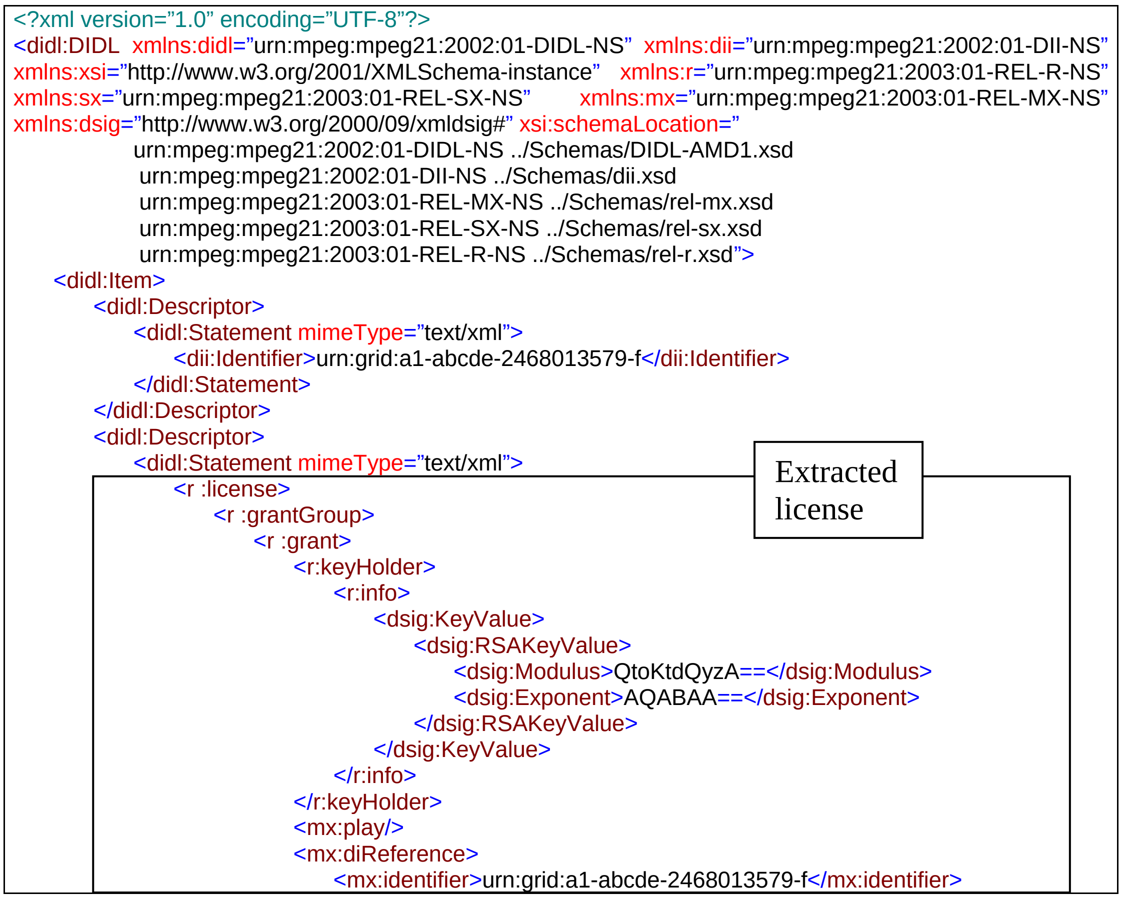
Next, we provide some image captions which show the inputs and results of Test 1.



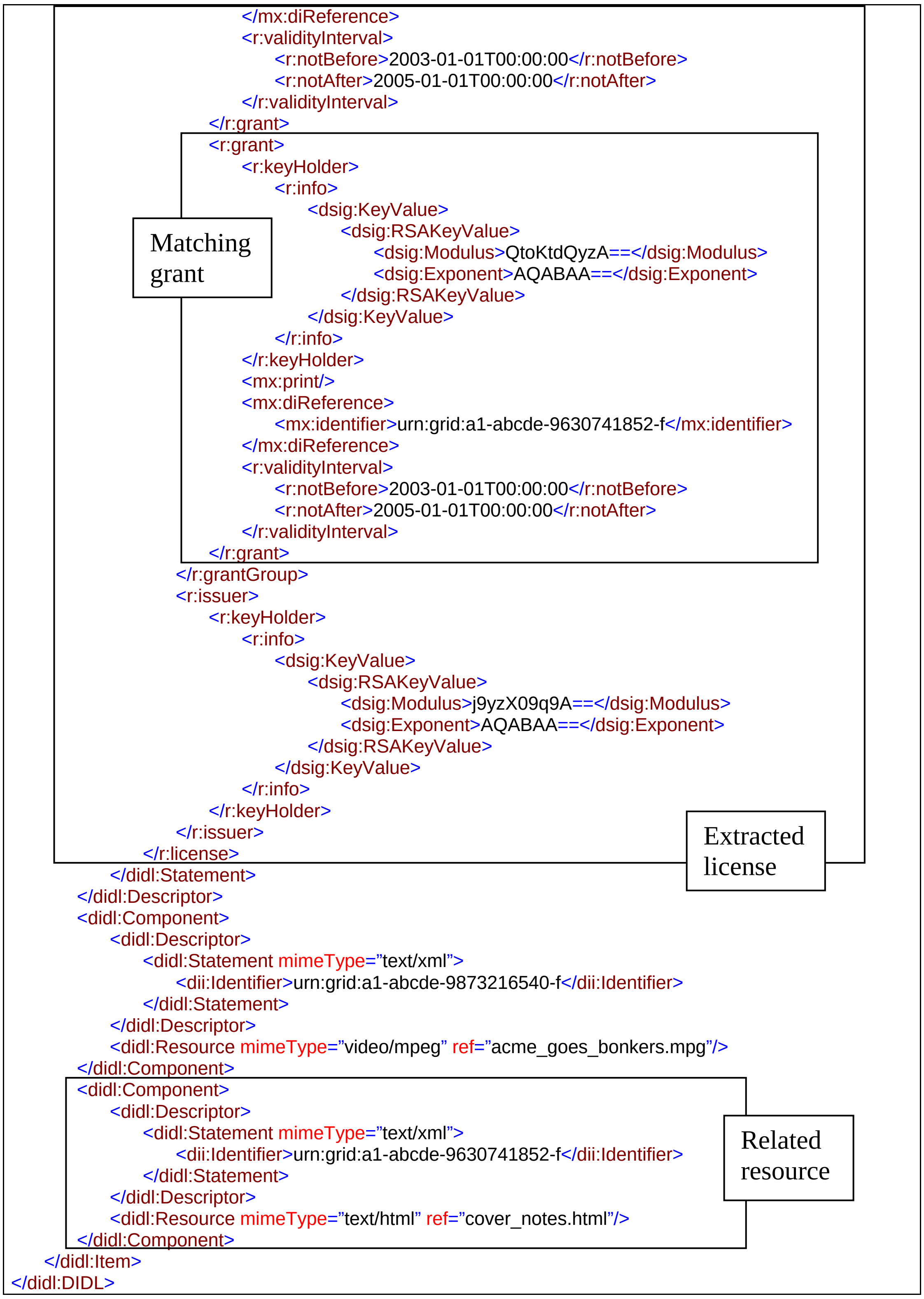
```
<goalToCheck>
    <principalToCheck>
        <modulusToCheck>QtoKtdQyzA==</modulusToCheck>
        <exponentToCheck>AQABAA==</exponentToCheck>
    </principalToCheck>
    <rightToCheck>mx:print</rightToCheck>
    <timeToCheck>2003-11-20T12:00:00</timeToCheck>
    <resourceToCheck>urn:grid:a1-abcde-9630741852-f</resourceToCheck>
</goalToCheck>
```

user
right
condition
resource

Figure 29. User Query in Test 1.



```
<?xml version="1.0" encoding="UTF-8"?>
<didl:DIDL  xmlns:didl="urn:mpeg:mpeg21:2002:01-DIDL-NS"  xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"   xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
xmlns:sx="urn:mpeg:mpeg21:2003:01-REL-SX-NS"        xmlns:mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" xsi:schemaLocation="
            urn:mpeg:mpeg21:2002:01-DIDL-NS ../Schemas/DIDL-AMD1.xsd
            urn:mpeg:mpeg21:2002:01-DII-NS ../Schemas/dii.xsd
            urn:mpeg:mpeg21:2003:01-REL-MX-NS ../Schemas/rel-mx.xsd
            urn:mpeg:mpeg21:2003:01-REL-SX-NS ../Schemas/rel-sx.xsd
            urn:mpeg:mpeg21:2003:01-REL-R-NS ../Schemas/rel-r.xsd">
    <didl:Item>
        <didl:Descriptor>
            <didl:Statement mimeType="text/xml">
                <dii:Identifier>urn:grid:a1-abcde-2468013579-f</dii:Identifier>
            </didl:Statement>
        </didl:Descriptor>
        <didl:Descriptor>
            <didl:Statement mimeType="text/xml">
                <r :license>
                    <r :grantGroup>
                        <r :grant>
                            <r:keyHolder>
                                <r:info>
                                    <dsig:KeyValue>
                                        <dsig:RSAKeyValue>
                                            <dsig:Modulus>QtoKtdQyzA==</dsig:Modulus>
                                            <dsig:Exponent>AQABAA==</dsig:Exponent>
                                        </dsig:RSAKeyValue>
                                    </dsig:KeyValue>
                                </r:info>
                            </r:keyHolder>
                            <mx:play/>
                            <mx:diReference>
                                <mx:identifier>urn:grid:a1-abcde-2468013579-f</mx:identifier>
```



```
                     </mx:diReference>
                     <r:validityInterval>
                        <r:notBefore>2003-01-01T00:00:00</r:notBefore>
                        <r:notAfter>2005-01-01T00:00:00</r:notAfter>
                     </r:validityInterval>
                  </r:grant>
                  <r:grant>
                     <r:keyHolder>
                        <r:info>
                           <dsig:KeyValue>
                              <dsig:RSAKeyValue>
                                 <dsig:Modulus>QtoKtdQyzA==</dsig:Modulus>
                                 <dsig:Exponent>AQABAA==</dsig:Exponent>
                              </dsig:RSAKeyValue>
                           </dsig:KeyValue>
                        </r:info>
                     </r:keyHolder>
                     <mx:print/>
                     <mx:diReference>
                        <mx:identifier>urn:grid:a1-abcde-9630741852-f</mx:identifier>
                     </mx:diReference>
                     <r:validityInterval>
                        <r:notBefore>2003-01-01T00:00:00</r:notBefore>
                        <r:notAfter>2005-01-01T00:00:00</r:notAfter>
                     </r:validityInterval>
                  </r:grant>
               </r:grantGroup>
               <r:issuer>
                  <r:keyHolder>
                     <r:info>
                        <dsig:KeyValue>
                           <dsig:RSAKeyValue>
                              <dsig:Modulus>j9yzX09q9A==</dsig:Modulus>
                              <dsig:Exponent>AQABAA==</dsig:Exponent>
                           </dsig:RSAKeyValue>
                        </dsig:KeyValue>
                     </r:info>
                  </r:keyHolder>
               </r:issuer>
            </r:license>
         </didl:Statement>
      </didl:Descriptor>
      <didl:Component>
         <didl:Descriptor>
            <didl:Statement mimeType=”text/xml”>
               <dii:Identifier>urn:grid:a1-abcde-9873216540-f</dii:Identifier>
            </didl:Statement>
         </didl:Descriptor>
         <didl:Resource mimeType=”video/mpeg” ref=”acme_goes_bonkers.mpg”/>
      </didl:Component>
      <didl:Component>
         <didl:Descriptor>
            <didl:Statement mimeType=”text/xml”>
               <dii:Identifier>urn:grid:a1-abcde-9630741852-f</dii:Identifier>
            </didl:Statement>
         </didl:Descriptor>
         <didl:Resource mimeType=”text/html” ref=”cover_notes.html”/>
      </didl:Component>
   </didl:Item>
</didl:DIDL>
```

Figure 30. DIDL document in Test 1.

```
Obtaining SchemaLocation…
for $c in document(“file:///C:\DMAGREL-DIDTool\Examples/did_urn.xml”)
return $c//@*:schemaLocation
End obtaining SchemaLocation.
Obtaining resource to check…
```

```
for $c in document("file:///C:\DMAGREL-DIDTool\Examples/user_urn.xml")
return $c//resourceToCheck/text()
Resource to check: urn:grid:a1-abcde-9630741852-f
End obtaining resource to check.

Obtaining URL…
for $c in document("file:///C:\DMAGREL-DIDTool\Examples/did_urn.xml")//*:Component
where ($c/*:Descriptor/*:Statement/*:Identifier/text()="urn:grid:a1-abcde-9630741852-f")
return $c/*:Resource/@ref
URL:  ref="cover_notes.html"
End obtaining URL.
Corrected URL: cover_notes.html

Resource: cover_notes.html
Identifier: urn:grid:a1-abcde-9630741852-f

Looking for licenses…
for $c in document("file:///C:\DMAGREL-DIDTool\Examples/did_urn.xml")//*:Statement
where     (($c//*:grant/*:diReference/*:identifier/text()="urn:grid:a1-abcde-9630741852-f")     or
($c//*:digitalResource/*:nonSecureIndirect[@URI="cover_notes.html"]))

return $c//*:licenseGroup
No matching licenseGroups found

for $c in document("file:///C:\DMAGREL-DIDTool\Examples/did_urn.xml")//*:Statement
where     (($c//*:grant/*:diReference/*:identifier/text()="urn:grid:a1-abcde-9630741852-f")     or
($c//*:digitalResource/*:nonSecureIndirect[@URI="cover_notes.html"]))
return $c//*:license

Writing license to outFiles/LicenseFile.xml…
End writing license.
End looking for licenses.

Executing REL License Interpreter…
User authorised
```

Figure 31. Command line caption in Test 1.

4.2.1.6 Open issue

Consider the above example, where in the DID document (outside the `license`) there is a `Component` that relates the resource in url format with the resource in urn format (resource identifier):

```
<didl:Component>
        <didl:Descriptor>
            <didl:Statement mimeType="text/xml">
                <dii:Identifier>urn:grid:a1-abcde-9630741852-f</dii:Identifier>
            </didl:Statement>
        </didl:Descriptor>
        <didl:Resource mimeType="text/html" ref="cover_notes.html"/>
</didl:Component>
```

Figure 32. Resource in url format and url format (resource identifier).

Then, if the User Query contains the resource to check in url format, the License Extractor is capable of finding a grant referred to the same resource in urn format (resource identifier), but the License Interpreter won't be able to perform the correct authorisation the binding information between the resource and its identifier is nowhere in the license.

```
<goalToCheck>
    <principalToCheck>
        <modulusToCheck>QtoKtdQyzA==</modulusToCheck>
        <exponentToCheck>AQABAA==</exponentToCheck>
    </principalToCheck>
    <rightToCheck>mx:print</rightToCheck>
    <timeToCheck>2003-11-20T12:00:00</timeToCheck>
    <resourceToCheck>cover_notes.html</resourceToCheck>
</goalToCheck>
```

Figure 33. User query.

```
<r :license>
    <r :grantGroup>
        <r :grant>
            <r:keyHolder>
                <r:info>
                    <dsig:KeyValue>
                        <dsig:RSAKeyValue>
                            <dsig:Modulus>QtoKtdQyzA==</dsig:Modulus>
                            <dsig:Exponent>AQABAA==</dsig:Exponent>
                        </dsig:RSAKeyValue>
                    </dsig:KeyValue>
                </r:info>
            </r:keyHolder>
            <mx:play/>
            <mx:diReference>
                <mx:identifier>urn:grid:a1-abcde-2468013579-f</mx:identifier>
            </mx:diReference>
            <r:validityInterval>
                <r:notBefore>2003-01-01T00:00:00</r:notBefore>
                <r:notAfter>2005-01-01T00:00:00</r:notAfter>
            </r:validityInterval>
        </r:grant>
        <r:grant>
            <r:keyHolder>
                <r:info>
                    <dsig:KeyValue>
                        <dsig:RSAKeyValue>
                            <dsig:Modulus>QtoKtdQyzA==</dsig:Modulus>
                            <dsig:Exponent>AQABAA==</dsig:Exponent>
                        </dsig:RSAKeyValue>
                    </dsig:KeyValue>
                </r:info>
            </r:keyHolder>
            <mx:print/>
            <mx:diReference>
                <mx:identifier>urn:grid:a1-abcde-9630741852-f</mx:identifier>
            </mx:diReference>
            <r:validityInterval>
                <r:notBefore>2003-01-01T00:00:00</r:notBefore>
                <r:notAfter>2005-01-01T00:00:00</r:notAfter>
            </r:validityInterval>
        </r:grant>
    </r:grantGroup>
    <r:issuer>
        <r:keyHolder>
            <r:info>
                <dsig:KeyValue>
                    <dsig:RSAKeyValue>
                        <dsig:Modulus>j9yzX09q9A==</dsig:Modulus>
                        <dsig:Exponent>AQABAA==</dsig:Exponent>
                    </dsig:RSAKeyValue>
                </dsig:KeyValue>
            </r:info>
        </r:keyHolder>
    </r:issuer>
</r:license>
```

Figure 34. Extracted license.

Figure 34 presents the extracted license. Note that the resource format, i.e. urn:grid:a1-abcde-2468013579-f, is not the same as in the user query, i.e. cover_notes.html, and the binding information is lost.

To solve this issue, we propose to modify the temporal LicenseFile file by changing the resource format so that the License Interpreter can match it with the resource in the same format as it appears in the User Query. This solution is presented in next section, in the enhanced version of the DID, REL and RDD integration.

4.2.2 Enhanced version of the DID, REL and RDD integration

4.2.2.1 Introduction

The second contribution to the DID-REL integration has been materialised in the "Contribution to DID/REL/RDD reference software: DMAG REL License Interpretation within DID using RDD term genealogy" [51] contribution to MPEG-21, which has been implemented as described in the MPEG-21 REL/RDD Software Implementation Plan [52], where it is included. It consists in an enhancement of the first contribution explained in section 4.2.1 and incorporates the RDD Ontology as a new feature.

This contribution integrates the previous DMAG License Extractor [50] with a DID Parser and the DMAG REL License Interpreter v1.2 [56] together with the RDD Ontology (RDDOnto) [57] through the RDDOnto API.

4.2.2.2 Architecture

The software implementation consists of three main modules: a DID Parser, the License extractor, and the already implemented "DMAG REL license interpreter using RDD term genealogy – implementation with web services" updated with the new DMAG License Interpreter v1.2. Some improvements have been done to the previous License Extractor as explained in section 4.2.2.3. Next, we describe the integrating modules.

The DID parser used in the software is a DMAG enhancement of the DID parser from Ghent University [58], which includes the DID Extensions in the AMD1 and the validation rules), improved to support the namespace prefixes. As the original parser did not allow using namespace prefixes, part of this research work has consisted on modifying its source code in order to support this enhanced and very important functionality. Figure 35 shows the structure of our solution, that we have named "DMAG REL license interpretation within DID using RDD term genealogy".

First of all, the DID document is parsed by the DID Parser to check its validity. If it is not valid, the reasons why are printed in the Log file. Otherwise, in the DMAG License extractor, the resource (in url or urn format) is extracted from the User Query file. Then, the DID document is traversed to identify the resource. If the resource in the Query is in url format (for instance, cover_notes.html), a urn identifier is sought. If the resource in the query is in urn identifier format (for instance, urn:grid:a1-abcde-9630741852-f), a url that matches the same resource is sought. Finally, the DID document is traversed to find all the licenses that match the resource (in both formats: url and urn). If a matching `license` or `licenseGroup` is obtained, then a temporal LicenseFile file is created, the DMAG REL license interpreter using RDD term genealogy is invoked and an Output file will be given as the output. Otherwise, the License Extractor directly gives the Output file. Refer to [53] and [56] for further information on the DMAG REL license interpreter using RDD term genealogy architecture and operation.

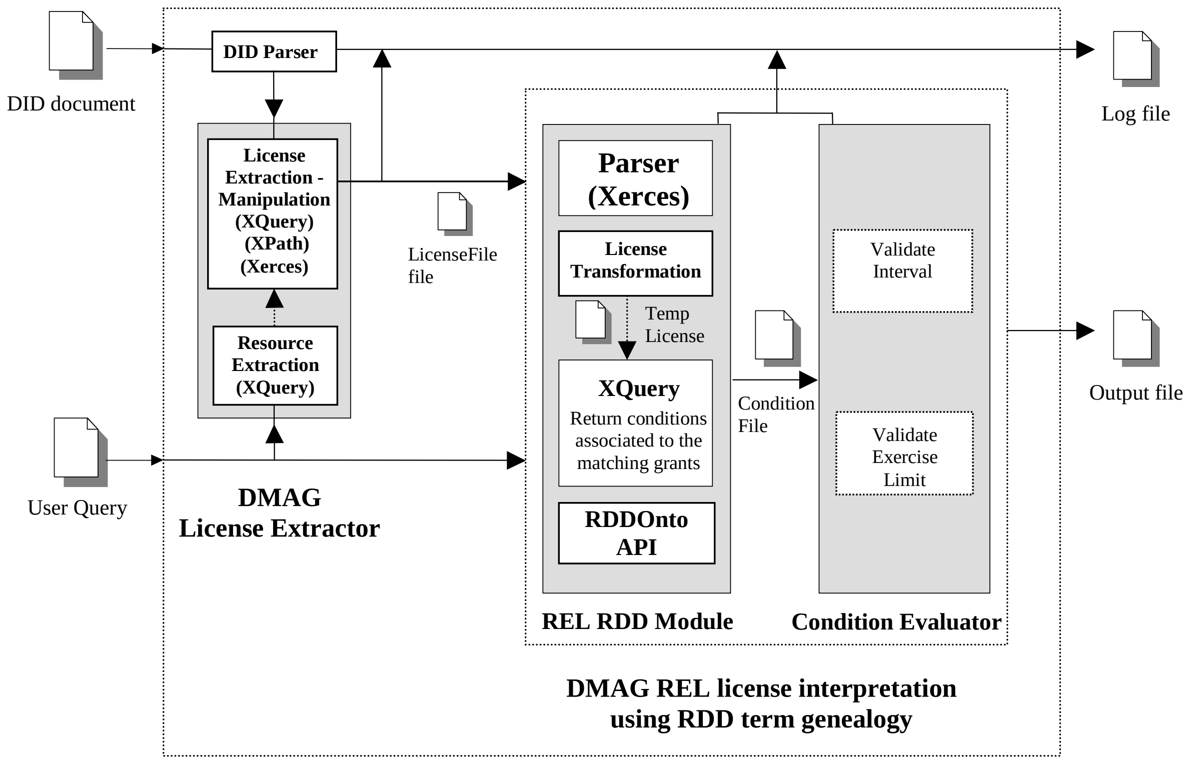


Figure 35. DMAG REL license interpretation within DID using RDD term genealogy.

4.2.2.3 Improvements done to this software with respect to the first version

Apart from the inclusion of a DID parser, and the integration together with the RDD functionality, this contribution to the DID, REL and RDD integration solves the initial problem explained in section 4.2.1.6.

To overcome the problem presented in section 4.2.1.6 we have modified the internal temporal LicenseFile file by changing the resource format so that the License Interpreter can match it with the resource in the same format as it appears in the User Query.

```
<goalToCheck>
    <principalToCheck>
        <modulusToCheck>QtoKtdQyzA==</modulusToCheck>
        <exponentToCheck>AQABAA==</exponentToCheck>
    </principalToCheck>
    <rightToCheck>mx:print</rightToCheck>
    <timeToCheck>2003-11-20T12:00:00</timeToCheck>
    <resourceToCheck>cover_notes.html</resourceToCheck>
</goalToCheck>
```

Figure 36. User query.

```
<r:grant>
    <r:keyHolder>
        <r:info>
            <dsig:KeyValue>
                <dsig:RSAKeyValue>
                    <dsig:Modulus>QtoKtdQyzA==</dsig:Modulus>
                    <dsig:Exponent>AQABAA==</dsig:Exponent>
                </dsig:RSAKeyValue>
            </dsig:KeyValue>
        </r:info>
    </r:keyHolder>
    <mx:print/>
    <mx:diReference>
        <mx:identifier>urn:grid:a1-abcde-9630741852-f</mx:identifier>
    </mx:diReference>
```

```
        <r:validityInterval>
            <r:notBefore>2003-01-01T00:00:00</r:notBefore>
            <r:notAfter>2005-01-01T00:00:00</r:notAfter>
        </r:validityInterval>
    </r:grant>
```

Figure 37. Matching grant in the extracted license. Note that the resource format (urn:grid:a1-abcde-2468013579-f) is not the same as in the user query (cover_notes.html) and the binding information in the DIDL document (see section 4.2.1.6) is lost.

```
<r:grant>
    <r:keyHolder>
        <r:info>
            <dsig:KeyValue>
                <dsig:RSAKeyValue>
                    <dsig:Modulus>QtoKtdQyzA==</dsig:Modulus>
                    <dsig:Exponent>AQABAA==</dsig:Exponent>
                </dsig:RSAKeyValue>
            </dsig:KeyValue>
        </r:info>
    </r:keyHolder>
    <mx:print/>
    <r:digitalResource>
        <r:nonSecureIndirect URI="cover_notes.html"/>
    </r:digitalResource>
    <r:validityInterval>
        <r:notBefore>2003-01-01T00:00:00</r:notBefore>
        <r:notAfter>2005-01-01T00:00:00</r:notAfter>
    </r:validityInterval>
</r:grant>
```

Figure 38. Matching grant in the modified temporal license. Note that now the resource format (cover_notes.html) is the same as in the user query.

4.2.2.4 Structure of the software

The software implementation that we propose consists of three main modules:

DID Parser

It checks the validity of the DID. It is a DMAG enhancement of the DID parser from Ghent University (including the DID Extensions in the AMD1 and the validation rules) made to support the namespace prefixes.

- Input:
    - DID instance document.
- Outputs:
    - DID tree.
    - A Log file if the DID is not valid, explaining the reasons why.

License Extractor

It obtains the resource from the user's query and looks for matching grants in the DID.

- Inputs**:**
    - A well formed DID document. Every license or licenseGroup must be a Statement child. For the moment, all grants must have a diReference or a nonSecureIndirect element to identify the resource they match to.
    - A User Query. For the moment, the resource must be in urn or url format.

- Outputs:
    - If any matching grants are found, the output is a LicenseFile file which contains the license(s) with the grants that match the resource.
    - If no matching grants are found, the output is an Output file saying that no matching grants were found.

REL License Interpreter using RDD term genealogy

Refer to [53] and [56] for further information on the DMAG REL License Interpreter using RDD term genealogy architecture and operation.

- Inputs:
    - A License.
    - A User Query.
- Outputs:
    - If there are any conditions associated to the matching grants, the output is an output file saying whether the conditions are satisfied or not.
    - If there are no conditions associated to the matching grants, the output is an output file saying there are no conditions associated to the matching grants.
    - A Log file if any problem occurs.

4.2.2.5 Software Details

- Languages: Java, XML Query, Xpath.
- Parser: Xerces DOM. Used in the DID Parser, License Interpreter using RDD term genealogy and License extractor to modify the temporal license.
- Supported Platforms: Windows, Linux.

4.2.2.6 Available tests

Several tests that illustrate the software operation using different DID document and User Query files have been created and provided together with the software. Moreover, an online demo is available at http://dmag.upf.edu/DIDTool/index.htm.

Table 7 shows the main features of each test. In the User Query file the resource is expressed in urn or url format. In the license, the resource can be expressed in the same format or in a different way, as explained in section 4.2.1.6. In the DID document, REL Namespaces and REL Schema Location definitions can be defined either in the DIDL, license or licenseGroup tag. In tests 1 to 8 no licenseGroup element is present whereas in tests 9 and 10 there is a single licenseGroup. Tests 1 to 15 show different cases where the right in the user query may or may not match the one in the license so that the RDD Onto API may be invoked. In tests 1 to 15, the Principal, Resource, Right and Condition to check from the User Query may or may not match those in the DID document. Tests 16 and 17 use invalid DID documents to show the operation of the DID Parser.

Table 7. Sample test main characteristics.

	Resource format		DID document			Right		Matching DID document – User Query				User authorised
Test	License	User Query	REL Namespaces and REL Schema Location definitions	LicenseGroup element	Valid	License	User Query	Principal	Resource	Right	Condition	
1	urn		`DIDL` tag	NO	YES	Print		YES	YES	YES	YES	YES
2	url		`license` tag	NO	YES	Print		YES	YES	YES	YES	YES
3	urn	url	`DIDL` tag	NO	YES	Print		YES	YES	YES	YES	YES
4	url	urn	`license` tag	NO	YES	Print		YES	YES	YES	YES	YES
5	urn		`DIDL` tag	NO	YES	Print		NO	YES	YES	YES	NO
6	urn		`DIDL` tag	NO	YES	Print		YES	NO	YES	YES	NO
7	url		`license` tag	NO	YES	Print	Play	YES	YES	NO	YES	NO
8	url		`license` tag	NO	YES	Print		YES	YES	YES	NO (time to check)	NO
9	urn		`DIDL` tag	YES	YES	Play Print	Play	YES	YES	YES	YES	YES
10	url		`licenseGroup` tag	YES	YES	Play Print	Print	YES	YES	YES	NO (time to check)	NO
11	url		`license` tag	NO	YES	Adapt	Play	YES	YES	YES	YES	YES
12	url		`license` tag	NO	YES	Adapt	Play	YES	YES	YES	NO (user limit > 5*)	NO
13	url		`license` tag	NO	YES	Adapt	Move	YES	YES	YES	YES	NO
14	url		`license` tag	NO	YES	Modify	Adapt	YES	YES	YES	YES	NO
15	urn		`license` tag	NO	YES	Modify	Move	YES	YES	YES	YES	YES
16	url		`license` tag	NO	NO	Play		YES	YES	YES	YES	NO
17	url		`license` tag	NO	NO	Play		YES	YES	YES	YES	NO

*The user limit condition present in the user query is not explicitly expressed in the DID Document but checked against a limit number given by a server.

4.3 Integration of DID, REL, RDD and DIP

4.3.1 Introduction

Based on the results that were achieved with the work on the integration of DID, REL and RDD, the second step that was considered was to go one step further and analyse the implications of an integration at the processing level. In this sense, the analysis of the Digital Item Processing Part (DIP) in MPEG-21 was a good means to, on one hand, to cover a new part of the standard and contribute to it with the DMAG expertise on different aspects mainly related to rights expression languages, and, on the other hand, to acquire, through the results obtained from a practical approach, a good knowledge of the requirements and implications of working at a processing level, which could help later in the definition of the general architecture for the management and protection of multimedia content.

After the analysis of the DIP part, we detected some inconsistencies in the edition of this part related to the rights enforcement at the processing level, which leaded to a discussion with the MPEG-21 DIP participants and to the final decision to perform a Core Experiment to show the different alternatives that could be adopted. The results of the Core Experiment on DIBOs for REL show how DID, DIP and REL and RDD can be integrated in the DIP context, present our implementation and experimental results for the proposed alternatives, and determine the advantages and disadvantages of creating specific DIBOs to implement the desired functionalities. The results of the core experiment have been contributed to MPEG-21 as an input document “MPEG-21 DIP Core Experiments: A contribution to the implementation of DIBOs for REL” [59], which includes a demonstration software tool.

This Core Experiment has leaded to the inclusion of two new DIBO interfaces in the edition of the DIP part of MPEG-21: GetLicense and QueryUserAuthorization DIBOs.

4.3.2 Problem statement

The DIP SoCD stated “*The Peer must ensure that rights, if present, are evaluated and enforced. This rights related functionality is not intended to be implemented within Digital Item Methods (by the DIM author), but instead must be implemented by the Peer (by the Peer vendor). Hence it is a requirement on the DIBO implementations that they execute according to the rights that are associated with the Digital Item under consideration*” and “*This part of ISO/IEC 21000 does not specify any tools at the DID level specifically for retrieving, evaluating and enforcing rights. However, implementation of DIBOs might have requirements or choices of implementation related to other parts of ISO/IEC 21000. For example, DIBO implementations might check for permissions, and in so doing, may take advantage of information compliant with ISO/IEC 21000-5 and 21000-6. DIBO implementations may make use of information specified by ISO/IEC 21000-7, if appropriate to the DIBO semantics*”.

However, the defined DIBOs that could have any relationship with rights enforcement, as for example PlayResource DIBO, did not specify any syntax or semantics on checking REL information at all, which leaded to confusion. The planned Core Experiment that we will present in next sections will analyse different alternatives of rights enforcement within DIP. The main difference between them resides in the place where the rights check is implemented. Whereas in the first and second alternatives the rights enforcement is located at the DIM level, in the third one the enforcement is located at the DIBO level.

4.3.3 Implemented DIBOs

In the core experiment report, only two DIBO interfaces have been investigated: GetLicense and RELUserAuthorization. According to the workplan for core experiment on DIBOs for REL [60] and after taking into account the discussions in the DIP e-mail reflector subsequent to our initial proposal that was distributed in the mentioned e-mail reflector, we have decided to implement the following DIBOs for applying REL practice within the DIP context:

4.3.3.1 GetLicense DIBO

The syntax of the GetLicense DIBO is the following:

Table 8. GetLicense DIBO syntax.

Syntax:	GetLicense (resourceNode)
Parameters:	resourceNode A MpegDIDNode object. It must be the MpegDIDNode correspondent to the DIDL Resource element over which the user wants to exercise a right
Return value:	A MpegDIDNode that contains the license that grants any rights to the correspondent resource.

This DIBO implements the extraction of the embedded licenses from the DID and optionally tries to retrieve them from an external server if any license is found in the DID.

Our current implementation is based on the License extractor presented in the REL license interpretation within DID using RDD term genealogy [51] which is part of the REL/RDD software implementation plan. The License extractor only extracts the rights expressions embedded in the DID that grant any right to the requested resource.

According to [61], it is upon the REL License Interpreter (RELUserAuthorization DIBO) to decide whether a right a user wants to exercise is granted by the license, so the GetLicense DIBO does not check any rights matching.

4.3.3.2 RELUserAuthorization DIBO

The syntax of the RELUserAuthorization DIBO is included in Table 9.

Table 9. RELUserAuthorization DIBO syntax.

Syntax:	RELUserAuthorization (licenseNode, right, resourceNode)
Parameters:	licenseNode The MpegDIDNode object correspondent to the license that may grant a user to exercise a right over a resource
Parameters:	right A String that contains the right the user wants to exercise resourceNode A MpegDIDNode object correspondent to the DIDL Resource element over which the user wants to exercise a right
Return value:	A Boolean value set to *true* if the user is authorised or *false* if not.

This DIBO checks if a user can exercise a right over a resource. The authorisation is based on the License information.

Our current implementation is based on REL/RDD reference software modules as the REL Schema Checker [62], REL Validation Rules Checker [63] and REL License Interpretation using RDD term genealogy.

This module first validates syntactically and semantically a License. It checks if the license is valid against the REL schemas and then validates the rules defined in the REL standard (21000-5). If the license is valid, it constructs an internal query with the user data extracted from a X.509 user certificate and the information that the DIBO is passed and executes it against the REL License. If no matching grants are found, it makes use of the MPEGOntosAPI [57] that invokes the operation getRDDSupertypes, which retrieves from the RDD ontology all the parents of the user right. Then, each of the parent rights is compared to the right in the License. If a matching grant is found the process finishes. Finally, the conditions associated to the matching grant are evaluated and if they are satisfied the user is authorised.

4.3.4 Analysed alternatives

Three alternatives have been analysed, according to the discussions that took place in the DIP e-mail reflector. The difference between them resides in the place where the rights check is implemented. Whereas in the first and second alternatives new DIBOs have been used to implement rights related functionalities (DIM-level rights enforcement), in the third one the PlayResource DIBO includes all these mechanisms (DIBO-level rights enforcement).

4.3.4.1 First alternative: license retrieval in a specific DIBO and rights check in the corresponding DIBO

In this case, there exists a specific DIBO that is responsible for retrieving a license, while the rights check is performed in the correspondent (e.g PlayResource) DIBO. An example of a DIM that plays a resource could be the following:

```
function PlayContent(Item){
        var resource=GetDIDLNode(“didl:Component/didl:Resource”, Item);
        if(resource!=null){
                var license=GetLicense(resource[0]);
                if(license!=null){ PlayResource(resource[0], license, false);}
                else Alert(“No license embedded in the DIDL document”, -1);
        }
}
```

Figure 39. PlayContent DIM – First approach.

4.3.4.2 Second alternative: license retrieval in a specific DIBO and rights check in a specific DIBO

In this case, there is a specific DIBO that is responsible for checking rights and another one that retrieves the license. An example of a DIM that plays a resource and uses the GetLicense DIBO could be the following:

```
function PlayContent(Item){
        var resource=GetDIDLNode(“didl:Component/didl:Resource”, Item);
        if(resource!=null){
                var license=GetLicense(resource[0]);
                if(license!=null){
                        if (RELUserAuthorization(license, “play”, resource[0])){
                                PlayResource(resource[0], false);
                        }
                        else Alert(“User is not authorised to play the resource”, -1);
                }
                else Alert(“No license embedded in the DIDL document”, -1);
        }
}
```

Figure 40. PlayContent DIM – Second approach.

4.3.4.3 Third alternative: license retrieval and rights check in the corresponding DIBO

In this case, there is not a specific DIBO that is responsible for retrieving a license nor for rights check. Instead, the license retrieval and rights check mechanisms are included in the PlayResource DIBO. An example of a DIM that plays a resource could be the following:

```
function PlayContent(Item){
        var resource=GetDIDLNode(“didl:Component/didl:Resource”, Item);
        if(resource!=null){
                PlayResource(resource[0], false);
        }
}
```

Figure 41. PlayContent DIM – Third approach.

4.3.5 Software architecture and modules

To perform the aforementioned Core Experiment, a software contribution has been developed. The software implementation consists in a GUI that permits the user to load a DID document by selecting the correspondent file. When a DI is selected, the LoadDID DIBO is called, which parses the DID document with the DID 2nd edition parser of Ghent University [58] and presents to the user the embedded DIMs. When a DIM is selected, the View DIM option becomes available so that the DIM code can be viewed. When a DIM is selected, if it has no input arguments, it can be directly executed. Otherwise, if it has any input arguments, the necessary arguments must be chosen from a list that is provided to the user before being able to execute the DIM. During the DIM execution, the necessary DIBOs are called and executed according to their implementation.

Three software packages are available in accordance with the three presented alternatives:

- The first one uses the GetLicense DIBO while the PlayResource DIBO implements the rights check.
- The second one uses the GetLicense DIBO and the RELUserAuthorization DIBO.
- The third one implements both the license retrieval and rights checks in the PlayResource DIBO.

4.3.6 Comparison between alternatives

The first alternative offers some advantages, as it keeps the license retrieval mechanism independent from the PlayResource DIBO. This makes it possible to easily change the license retrieval mechanisms, avoiding the existence of multiple PlayResource DIBOs that only differ in these mechanisms.

On one hand, checking rights in a specific DIBO (section 4.3.4.2) would avoid the need to update the full PlayResource DIBO whenever extensions to REL are made, needing only to update the RELUserAuthorization DIBO. If these specific DIBOs are used, DI authors should take the responsibility of controlling the user's access to the resources according to specific rights. A trusted device would respect the rights fulfilling.

On the other hand, the enforcement of rights expressions could be implemented in the correspondent DIBOs (e.g. PlayResource DIBO), as seen in section 4.3.4.3. As trusted implementations of DIBOs will be provided for users, the rights enforcement will be ensured without the need of controlling it in DIMs.

The two last alternatives are both valid, although 4.3.4.2 doesn't follow the specifications in clause 5.3.1 of the DIP SoCD [41]. However, the current PlayResource does not specify any syntax or semantics on checking REL information at all, which leads to confusion.

Another possibility could be to consider the interfaces of the proposed DIBOs as normative subroutines that could be used within other DIBOs.

4.3.7 Results and impact of the Core Experiment

After the presentation of the Core Experiment at the MPEG-21 Redmond meeting, two new DIBOs were included in the DIP part. These two DIBOs are GetLicense and QueryLicenseAuthorization and correspond to the GetLicense and RELUserAuthorization DIBOs analysed in the Core Experiment. Their syntax and interface is provided below:

Table 10. Standardised GetLicense DIBO.

Syntax :	`GetLicense( resource )`
Description :	Gets licenses associated with the given resource.
Parameters:	`resourceNode` The `Element` object that represents the DIDL RESOURCE element.
Return value:	An array of `Element` objects that represent any licenses or null if there is no license associated with the resource.
Exceptions:	None.

Table 11. Standardised QueryLicenseAuthorization DIBO.

Syntax:	`QueryLicenseAuthorization( license, resource, rightNs, rightLocal, additionalInfo )`
Description:	Checks for the existence of an authorization proof for an authorization request formed according to the semantics of this DIBO given below.
Parameters:	`license` The `Element` object that represents the license information.
	`Resource` The `Element` object that represents the DIDL RESOURCE element.
	`RightNs` The `String` object that represents the namespace of the right to be

	checked or `null`
	`rightLocal` The `String` object that represents the localname of the right to be checked or the value of the definition attribute of `sx:rightUri`, depending on whether `rightNs` is a `String` or `null`, respectively.
	`AdditionalInfo` An array containing `Element` objects representing additional information that can be considered when validating the license. This parameter may be null, in which case no additional information is provided by this parameter.
Return value:	Boolean value with value true if a corresponding authorization proof is found and false if a corresponding authorization proof does not exist or could not be found.
Exceptions:	None.

These two new DIBOs are not intended to enforce the rights accomplishment at the DIM level, but provide the implementer a means to access the license information or the authorisation results before performing the definitive action.

The GetLicense DIBO provides an interface for the DIM author to be able to retrieve license information. However, it is not intended that this DIBO be used to protect a resource. The User is expected to always check and enforce rights regardless of whether a Digital Item is interacted with via a DIM or not. In the case a Digital Item is interacted with via a DIM, the protection is performed by the underlying library of DIBO implementations.

Regarding the QueryLicenseAuthorization DIBO, implementations of this DIBO can knowingly construct a false authorisation context to try to meet this DIBO's goals of providing a preliminary screening of User intent before continuing on with the DIM. For example, if the license is conditioned upon a per-use payment of $3, the DIBO implementation might ask a human if he wants to pay $3 for the associated right on the associated resource. Then, without charging the user $3, the DIBO implementation might still use an authorisation context that says the user paid $3 so that the result value that gets returned reflects whether the user would be authorised if he were to pay $3. However, it is not intended that this DIBO be used to allow real access to a resource. The User is expected to always check and enforce rights regardless of whether a Digital Item is interacted with via a DIM or not.

4.4 Distributed generation of Event Reports

4.4.1 Introduction

MPEG-21 Event Reporting standard [31] provides a standardised means for sharing information about events, related to Digital Items or Peers that interact with them, amongst Peers and Users.

Event Reporting standard specification has defined some mechanisms to identify the peers that have created and/or modified and Event Report. For this purpose, the `Modification` element, which maintains the history of modifications of the ER, has been defined. This element consists of the `PeerId`, `UserId`, `Time` and `Description` elements. The `PeerId` element identifies the Peer that has created or modified the ER; the `UserId` element identifies the User that has created or modified the ER; the `Time` element specifies the date and time at which the ER was created of modified; and the `Description` element is a free form field to provide additional information.

On the other hand, the reported information is a placeholder for inclusion of the reported data into an ER. Figure 42 sketches the structure of an ER.

Therefore, in an Event Report we can specify the Peers that have created and modified the ER by means of different Modification elements, but we can not specify the fields or the data that each Peer has reported or modified in a structured way.

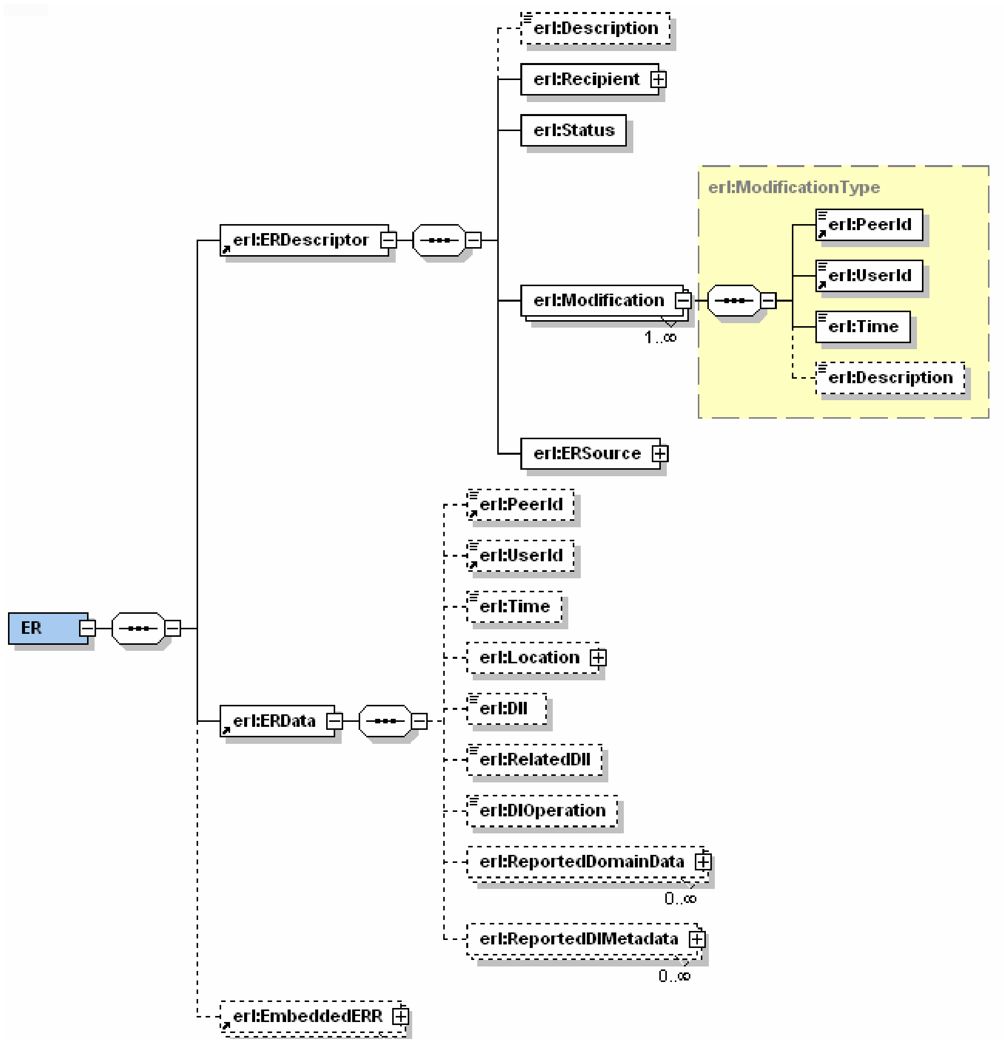


Figure 42. ER element.

In this section we will first show the limitations of current Event Report structure with regards to the modifications introduced by the different Peers that are involved in its generation, and the implications it has when trying to deal with secure Event Reports. Finally, we will propose an improvement to ISO/IEC 21000-15 [31] to solve the presented limitations.

This proposal has been contributed to MPEG-21 as an input document “Some issues on the generation and modification of Event Reports in the MPEG-21 Event Reporting” [64]. Moreover, the results of this work have been contributed as two papers to the following International conferences: the Fourth International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS’08) [73] and the 14th International Conference on Distributed Multimedia Systems (DMS

2008) [74]. The first paper is acceptance pending, while the latter is accepted and publication pending at the moment of writing.

4.4.2 Content Consumption Scenario

In this section we present a content consumption scenario, which illustrates how an Event Report could be generated and modified by different parties, before reaching its final destination.

The scenario refers to the case where an end user performs an action over an object that is protected and governed, that is, which includes a digital license expressing the rights and conditions of usage of the related content.

Figure 43 scenario involves a user that wants to e.g. play a song. This action attempt would be done by using a user-side tool and trusted module, which will be responsible for the enforcement of rights and generation of event reports. The trusted module could be an integral part of the tool or otherwise a plug-in for an already existing tool, which enables the consumption of digital objects that use the system specific packaging and protection format.

The sequence diagram would be as follows:

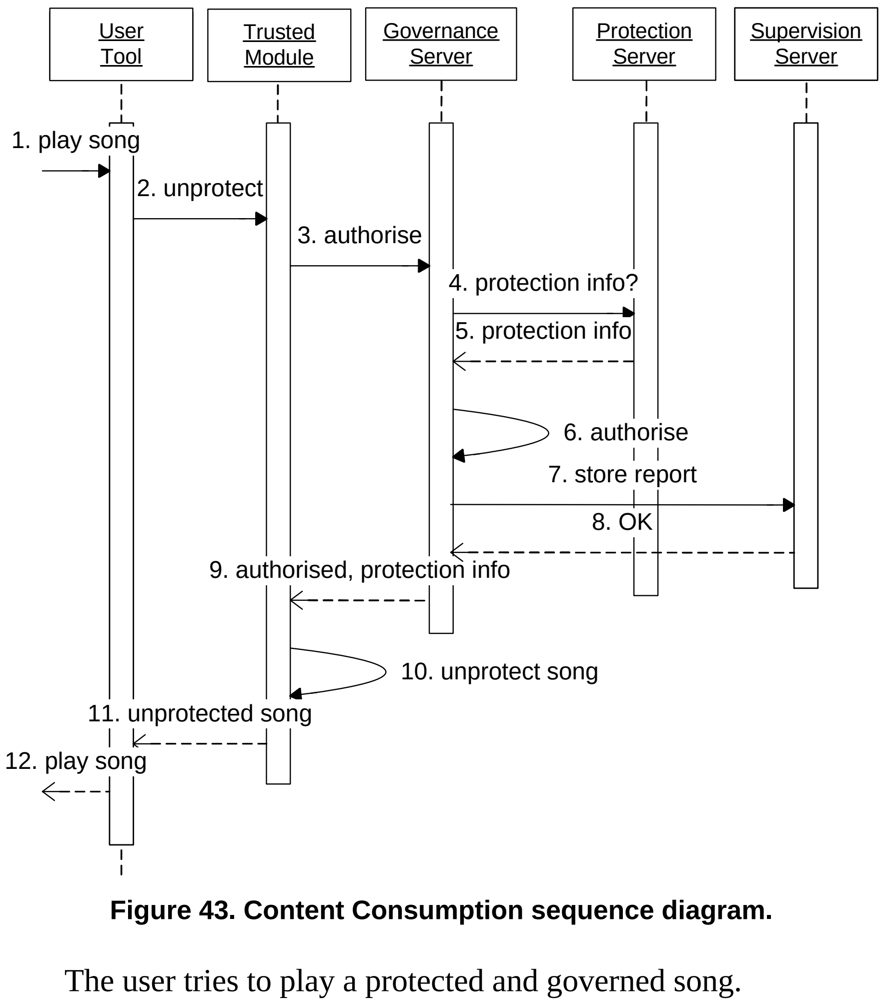


Figure 43. Content Consumption sequence diagram.

1. The user tries to play a protected and governed song.
2. The user tool asks for unprotection to the trusted module.
3. The trusted module requests authorisation to the Governance server and sends an Event Report which involves the user request.

4-5. The Governance server retrieves the protection information necessary to unprotect the object.
5-5. The Governance server performs the license-based authorisation using the licenses in its license database.
7-8. If the authorisation is successful, the Governance server modifies the Event Report received from the user to add the license identifier used for the authorisation and sends it to the Supervision server, which collects and stores it.
9. The Governance server returns the authorisation result and protection information to the trusted module.
10. The trusted module unprotects the song and gives the control to the user tool.

In step 7, the event report received from the user is modified by the Governance Server and sent to the Supervision server.

Figure 44 shows a typical ER created during content consumption. Note that we can not distinguish in the ER the specific data that has been reported by the Trusted Module and the modification made by the Governance server.

```
<?xml version="1.0" encoding="UTF-8"?>
<ER xmlns="urn:mpeg:mpeg21:2005:01-ERL-NS" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:mpeg:mpeg21:2005:01-
ERL-NS er.xsd">
  <ERDescriptor>
    <Recipient>
      <!—Supervision server ID -->
      <PeerId>urn:mipams:SID:1l4e28ba-2fa1-11i2-8j3l-j1a711die3fj</PeerId>
    </Recipient>
    <Status value="true"/>
    <Modification>
      <!-- End user modification-->
      <PeerId>urn:mipams:TID:4bla0c2-ce3q-1db7-7s5j-njfie92rh48v</PeerId>
      <UserId>urn:mipams:UID:3nkr14jg-2ih9-17fc-3fbf-nfi1i6jinvf1</UserId>
      <Time>2006-11-20T12:22:30</Time>
    </Modification>
    <Modification>
      <!-- Governance Server modification-->
      <PeerId>urn:mipams:GID:1jvrt5ba-2gr1-t46b-j56f-bn1ui57za41</PeerId>
      <UserId>urn:mipams:UID:3nkdsfjg-2idfs9-1dfc-3fuf-nf3rgt54vf2</UserId>
      <Time>2006-11-20T12:22:32</Time>
    </Modification>
    <ERSource>
      <OtherSource>urn:mipams:TID:4bla0c2-ce3q-1db7-7s5j-njfie92rh48v</OtherSource>
    </ERSource>
  </ERDescriptor>
  <ERData>
    <PeerId>urn:mipams:TID:4bla0c2-ce3q-1db7-7s5j-njfie92rh48v</PeerId>
    <UserId>urn:mipams:UID:3nkr14jg-2ih9-17fc-3fbf-nfi1i6jinvf1</UserId>
    <Time>2006-11-20T12:22:30</Time>
    <Location>
        <mpeg7:Region>es</mpeg7:Region>
    </Location>
    <!-- Object ID -->
    <DII>urn:mipams:OBJ:8b4fkt5a-2hq4-1gnm-8a2f-r9gnj157i8fb</DII>
    <DIOperation>REL:mx:Play</DIOperation>
    <ReportedDomainData>
      <Name>urn:mipams:DOM:19h6k2ba-46a1-1h62-8s31-b5hzxnwy667b</Name>
    </ReportedDomainData>
    <ReportedDIMetadata>
     <!-- Specific Metadata -->
     <CollectingSocietyId>urn:mipams:CSID:5f4e28ba-2fa1-61p2-lqf-2f7r1y687f8</CollectingSocietyId
```

```
      <CreatorId>urn:mipams:CID:6d4e28ga-2fai-18h2-3y3f-b95761dbe3fb</CreatorId>
      <WorkId>urn:mipams:WID:8j4e2fca-2fu1-1sb2-o83f-18a64ydhag4b</WorkId>
      <DistributorId>urn:mipams:DID:9b4e26ba-5fa1-15u2-583f-39a766qbe3fb</DistributorId>
      <!-- End User License ID -->
      <LicenseId>urn:mipams:LID:2c4eg8sa-2sa1-b182-h9d5-bhny41dh67fb</LicenseId>
      <ToolFingerprint>FhRuD1iGkUbej0fwBzT92Q==</ToolFingerprint>
    </ReportedDIMetadata>
  </ERData>
</ER>
```

Figure 44. ER Contents in Content Consumption scenario.

4.4.3 How to cope with modifications using ISO/IEC 21000-15

If we consider current ISO/IEC 21000-15 standard specification [31], we have two alternatives to indicate the data that have been reported or modified by a specific Peer within an ER.

The first option consists of including the fields that a Peer has reported in the `Description` element of the `Modification` element of an ER. The `Modification` element, as defined in ISO/IEC 21000-15, is a free form field to provide additional information and its type is `xsd:String`. By choosing this alternative, the information can not be clearly structured within the `Description` element. Figure 45 shows an example of an ER that uses `Description` element to indicate the elements reported by the different Peers involved in the creation and modification of an ER.

```
<?xml version="1.0" encoding="UTF-8"?>
<ER xmlns="urn:mpeg:mpeg21:2005:01-ERL-NS" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:mpeg:mpeg21:2005:01-
ERL-NS er.xsd">
  <ERDescriptor>
    <Recipient>
      <!-- Recipient Identification: Supervisor ID -->
      <PeerId>urn:mipams:SID:1l4e28ba-2fa1-11i2-8j3l-j1a711die3fj</PeerId>
    </Recipient>
    <Status value="true"/>
    <Modification>
      <!-- End user modification-->
      <PeerId>urn:mipams:TID:4bla0c2-ce3q-1db7-7s5j-njfie92rh48v</PeerId>
      <UserId>urn:mipams:UID:3nkr14jg-2ih9-17fc-3fbf-nfi1i6jinvf1</UserId>
      <Time>2006-11-20T12:22:30</Time>
      <Description>PeerId, UserId, DII, DIOperation, Location …</Description>
    </Modification>
    <Modification>
      <!-- Governance server modification-->
      <PeerId>urn:mipams:GID:1jvrt5ba-2gr1-t46b-j56f-bn1ui57za41</PeerId>
      <UserId>urn:mipams:UID:3nkdsfjg-2idfs9-1dfc-3fuf-nf3rgt54vf2</UserId>
      <Time>2006-11-20T12:22:32</Time>
       <Description>LicenseID</Description>
    </Modification>
    <ERSource>
      <OtherSource>urn:mipams:TID:4bla0c2-ce3q-1db7-7s5j-njfie92rh48v</OtherSource>
    </ERSource>
  </ERDescriptor>
  <ERData>
    <!-- Who, where and when-->
    <PeerId>urn:mipams:TID:4bla0c2-ce3q-1db7-7s5j-njfie92rh48v</PeerId>
    <UserId>urn:mipams:UID:3nkr14jg-2ih9-17fc-3fbf-nfi1i6jinvf1</UserId>
    <Time>2006-11-20T12:22:30</Time>
    <Location>
        <mpeg7:Region>es</mpeg7:Region>
    </Location>
    <!-- Involved Object, Operation and others-->
    <DII>urn:mipams:OBJ:8b4fkt5a-2hq4-1gnm-8a2f-r9gnj157i8fb</DII>
```

```
      <DIOperation>REL:mx:Play</DIOperation>
      <ReportedDomainData>
        <Name>urn:mipams:DOM:19h6k2ba-46a1-1h62-8s31-b5hzxnwy667b</Name>
      </ReportedDomainData>
      <ReportedDIMetadata>
        <!-- Specific Metadata -->
        <CollectingSocietyId>urn:mipams:CSID:5f4e28ba-2fa1-61p2-lqf-2f7r1y687f8</CollectingSocietyId
        <CreatorId>urn:mipams:CID:6d4e28ga-2fai-18h2-3y3f-b95761dbe3fb</CreatorId>
        <WorkId>urn:mipams:WID:8j4e2fca-2fu1-1sb2-o83f-18a64ydhag4b</WorkId>
        <DistributorId>urn:mipams:DID:9b4e26ba-5fa1-15u2-583f-39a766qbe3fb</DistributorId>
        <!-- End User License ID -->
        <LicenseId>urn:mipams:LID:2c4eg8sa-2sa1-b182-h9d5-bhny41dh67fb</LicenseId>
        <ToolFingerprint>FhRuD1iGkUbej0fwBzT92Q==</ToolFingerprint>
      </ReportedDIMetadata>
    </ERData>
</ER>
```

Figure 45. ER contents – Description element indicates the fields that each Peer has reported.

4.4.4 Proposal

This section presents three different proposals that will allow the specification of the data that each Peer has reported.

The first option we propose is to change the type of the `Description` child element of the `Modification` to `xsd:any`. In this way, we can describe in a structured way the elements in the `ERData` that each Peer has created or modified.

The second option consists in adding a new child element to the `Modification` that will be used to indicate the elements of an ER that a Peer has created and/or modified. This second option allows using the `Description` element as a free form field to provide additional information, just as it is currently defined in ISO/IEC 21000-15.

The third option consists in enabling the use of more than one `ERData` in an ER. Moreover, this third option enables the association of the Modification information, as the Peer or User that has created or modified the ER, with the specific data that this Peer or User has reported. For this purpose, we have defined the `idData` attribute for the `ERData` element to uniquely identify this element in an ER; and the `idRefData` attribute for the `Modification` element to reference to the data that a Peer or User has reported.

For the three options presented, only the last one will enable Users and Peers to digitally sign the data that they report. Then, we propose the adoption of this solution to enable specifying in an ER the data that each Peer or User has reported. Moreover, using this solution the Peers and Users can digitally sign the data reported.

4.4.5 ER element Proposal

This section presents the ER element that we propose (see Figure 46) and an example of use (see Figure 47).

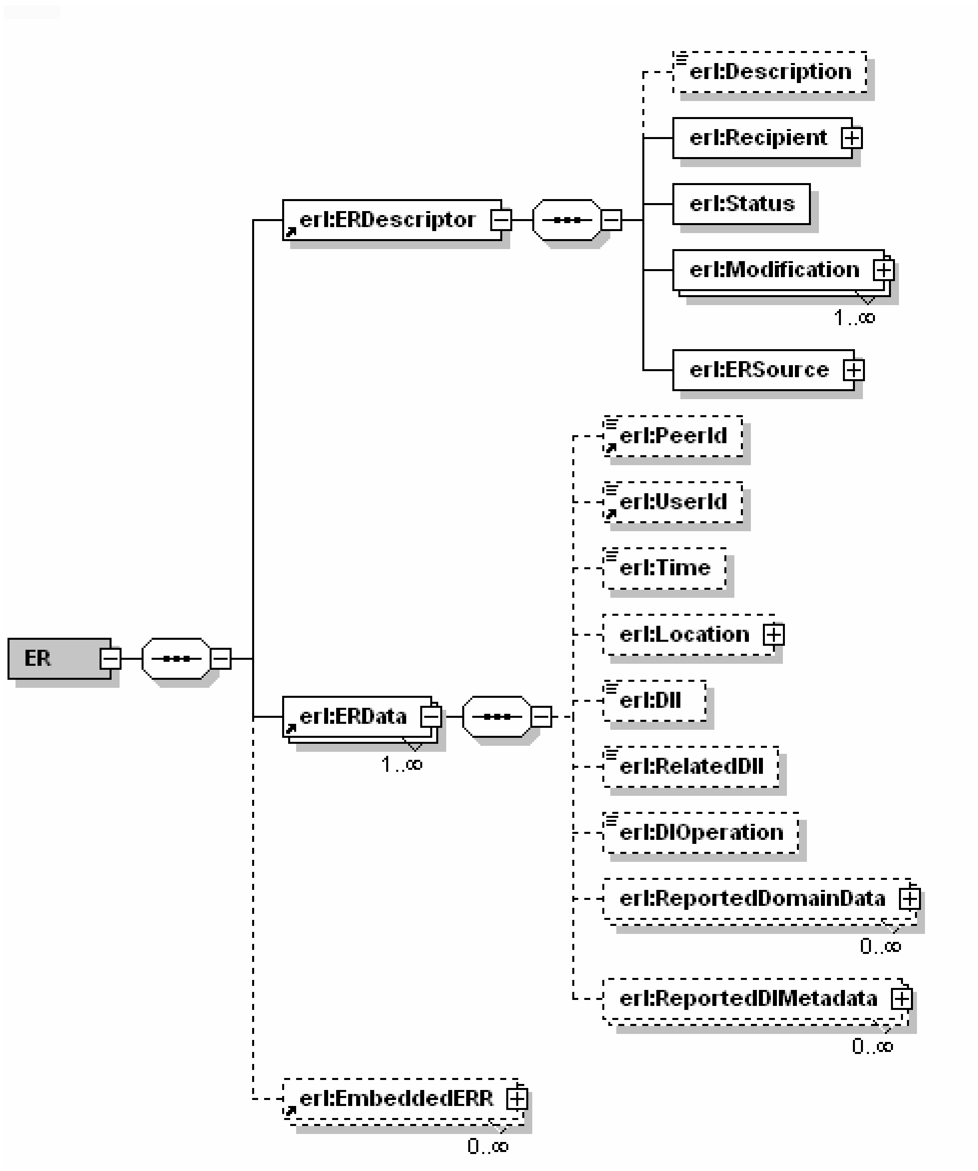


Figure 46. ER element.

```
<?xml version="1.0" encoding="UTF-8"?>
<ER xmlns="urn:mpeg:mpeg21:2005:01-ERL-NS" xmlns:mpeg7="…"
xmlns:dsig="…" xmlns:xsi="…" xsi:schemaLocation="…">
  <ERDescriptor>
    <!-- Recipient Identification: Supervision Server ID -->
    <Recipient>
      <PeerId>urn:mipams:SID:1l4e28ba-2fa1-11i2-8j3l-j1a711die3fj</PeerId>
    </Recipient>
    <Status value="true"/>
    <Modification idDataRef=”D001”>
      <!-- End user modification-->
      <PeerId>urn:mipams:TID:4bla0c2-ce3q-1db7-7s5j-njfie92rh48v</PeerId>
      <UserId>urn:mipams:UID:3nkr14jg-2ih9-17fc-3fbf-nfi1i6jinvf1</UserId>
      <Time>2006-11-20T12:22:30</Time>
      <Description>PeerId, UserId, DII, DIOperation, Location …</Description>
    </Modification>
    <Modification idDataRef=”D002”>
      <!-- Governance server modification-->
      <PeerId>urn:mipams:GID:1jvrt5ba-2gr1-t46b-j56f-bn1ui57za41</PeerId>
      <UserId>urn:mipams:UID:3nkdsfjg-2idfs9-1dfc-3fuf-nf3rgt54vf2</UserId>
      <Time>2006-11-20T12:22:32</Time>
```

```
         <Description>LicenseID</Description>
      </Modification>
      <!-- Source of the ER: User Tool Identifier -->
      <ERSource>
         <OtherSource>urn:mipams:TID:4bla0c2-ce3q-1db7-7s5j-njfie92rh48v</OtherSource>
      </ERSource>
   </ERDescriptor>
   <ERData idData="D001">
      <!-- Who, where and when-->
      <PeerId>urn:mipams:TID:4bla0c2-ce3q-1db7-7s5j-njfie92rh48v</PeerId>
      <UserId>urn:mipams:UID:3nkr14jg-2ih9-17fc-3fbf-nfi1i6jinvf1</UserId>
      <Time>2006-11-20T12:22:30</Time>
      <Location>
           <mpeg7:Region>es</mpeg7:Region>
      </Location>
      <!-- Involved Object, Operation and others-->
      <DII>urn:mipams:OBJ:8b4fkt5a-2hq4-1gnm-8a2f-r9gnj157i8fb</DII>
      <DIOperation>REL:mx:Play</DIOperation>
      <ReportedDomainData>
         <Name>urn:mipams:DOM:19h6k2ba-46a1-1h62-8s31-b5hzxnwy667b</Name>
      </ReportedDomainData>
      <ReportedDIMetadata>
        <!-- Specific Metadata -->
        <CollectingSocietyId>urn:mipams:CSID:5f4e28ba-2fa1-61p2-lqf-2f7r1y687f8</CollectingSocietyId
        <CreatorId>urn:mipams:CID:6d4e28ga-2fai-18h2-3y3f-b95761dbe3fb</CreatorId>
        <WorkId>urn:mipams:WID:8j4e2fca-2fu1-1sb2-o83f-18a64ydhag4b</WorkId>
        <DistributorId>urn:mipams:DID:9b4e26ba-5fa1-15u2-583f-39a766qbe3fb</DistributorId>
        <ToolFingerprint>FhRuD1iGkUbej0fwBzT92Q==</ToolFingerprint>
      </ReportedDIMetadata>
       <!-- User Digital Signature-->
         <dsig:Signature>
            <dsig:SignedInfo>
               <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
               <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
               <dsig:Reference>
                  <dsig:Transforms>
                     <dsig:Transform Algorithm="urn:mpeg:mpeg21:2007:01-ER-erTransform"/>
                     <dsig:Transform Algorithm="urn:uddi-org:schemaCentricC14N:2002-07-10"/>
                  </dsig:Transforms>
                  <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                  <!-- ER Digest -->
                  <dsig:DigestValue>P7aPcmFukghoi4y39gfcDFgov7H=</dsig:DigestValue>
               </dsig:Reference>
            </dsig:SignedInfo>
            <!-- Signed Digest -->
            <dsig:SignatureValue>pRj0rxmxWQEQhIIAYbtcIcmo8M=</dsig:SignatureValue>
            <dsig:KeyInfo>
               <dsig:KeyValue>
                  <dsig:RSAKeyValue>
                     <dsig:Modulus>0xR9lZdUEF0ThO4w==</dsig:Modulus>
                     <dsig:Exponent>AQABAA==</dsig:Exponent>
                  </dsig:RSAKeyValue>
               </dsig:KeyValue>
            </dsig:KeyInfo>
         </dsig:Signature>
   </ERData>
   <ERData idData="D002">
      <ReportedDIMetadata>
        <!-- End User License ID -->
        <LicenseId>urn:mipams:LID:2c4eg8sa-2sa1-b182-h9d5-bhny41dh67fb</LicenseId>
      </ReportedDIMetadata>
      <!-- Governance server Digital Signature-->
      <dsig:Signature>
         <dsig:SignedInfo>
            <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <dsig:Reference>
```

```
          <dsig:Transforms>
            <dsig:Transform Algorithm="urn:mpeg:mpeg21:2007:01-ER-erTransform"/>
            <dsig:Transform Algorithm="urn:uddi-org:schemaCentricC14N:2002-07-10"/>
          </dsig:Transforms>
          <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          <!-- ER Digest -->
          <dsig:DigestValue>Q6bNcmGachtoi3z65gcfFDsaf6X=</dsig:DigestValue>
        </dsig:Reference>
      </dsig:SignedInfo>
      <dsig:SignatureValue>pRj0rxmxWQEQhIIAYbtcIcmo8M=</dsig:SignatureValue>
      <dsig:KeyInfo>
        <dsig:KeyValue>
          <dsig:RSAKeyValue>
            <dsig:Modulus>0xQ8lXdARF0ThO4w==</dsig:Modulus>
            <dsig:Exponent>AQEBCC==</dsig:Exponent>
          </dsig:RSAKeyValue>
        </dsig:KeyValue>
      </dsig:KeyInfo>
    </dsig:Signature>
  </ERData>
</ER>
```

Figure 47. ER example.

4.5 Conclusions

In this chapter we have presented our work on the integration of MPEG-21 DID, REL, RDD, DIP and Event Reporting parts.

First of all, we have worked on the integration of DID and REL, which has involved the analysis of the DID and REL parts, in order to determine how licenses can be embedded or extracted in/from DIDL documents. Once this has been achieved, a license extraction mechanism has been implemented, thus enabling the integration with the already existing DMAG simple authorisation software. This integration has consisted on, given a DIDL document with an embedded license (or licenses) and a user query that expresses that a user intends to perform an action over a resource in the DIDL document, performing a simple authorisation, That is, it tries to answer the question: “Is the user A allowed to perform the right B over the resource C (included in the DIDL document) under the conditions D, E, etc. according to the terms expressed in any of the licenses embedded in the DIDL document?”.

Then, we have presented a second version, which includes the integration of DID, REL and RDD. This second contribution offers additional functionality, including a DID parser, the RDD functionality and some additional features that enhance the first version. The two contributions regarding the integration of different parts of the MPEG-21 standard have been included in the MPEG-21 standard in the reference software part [26].

After this, we have performed an integration at the processing level, involving DID, REL, RDD and DIP. This last integration shows how DIDL documents can be managed from a DIP perspective, and how authorisations can be performed in this context. DIP has been taken as the reference point that enables the relationships between the rest of the MPEG-21 parts. Regarding the DIP contribution, we have proposed the standardisation of the interfaces of two new elements in the DIP part to support some functionalities related to license retrieval and authorisation of users integrated in the DIP context. These two interfaces have been accepted and added to the DIP specification.

used to work in the definition of an architecture for the management and protection of multimedia information, as described in Chapter 4 of this document.

Finally, we have presented the proposal we have made to MPEG-21 Event Reporting in order to solve the problem that arises when reporting information coming from different sources, regarding which information has been added by which of the parties. This proposal is the result of the usage of the standard in the different architectures described in Chapter 4 of this document. This contribution has been considered, accepted and included in the resulting corrigendum [49], having a direct impact on the text of the standard.

5 MIPAMS: a generic architecture for the management and protection of multimedia information

5.1 Motivation

Multimedia information management, which involves all the steps of content lifecycle, from the creation and production to the distribution and consumption, is a complex and challenging research area. To have a secure and trusted system we need to take into account aspects such as digital rights management (DRM), certification, control and security. As current solutions rely on proprietary architectures and tools, or are not open or based on standards, we will define an open architecture, as general as possible and not restricted to a specific standard, which provides trust and rights management in multimedia information systems. To achieve this goal, all the experience of the DMAG group in the security and DRM area will be very valuable and the results obtained in the integration work that has been already presented in this document will be taken into account.

After a first approximation, mainly based on MPEG-21, we will present a more general architecture, which is not restricted to a sole standard. We will define the elements of the architecture needed to provide trust to the whole value chain by managing multimedia content and digital rights represented using current standards, such as MPEG-21 and OMA DRM, see how the standards can be mapped to it, and compare it with an alternative approach, the OpenSDRM architecture. It is worth noting that currently the architecture does not try to be complete, and it lacks some modules, such as a payment solution, which are deliberately ignored.

Finally, we will present more details concerning its implementation and usage in different projects such as the AXMEDIS Integrated Project [21], the VISNET II Network of Excellence [19] and the Spanish project GILDDA [20].

Each of the sections details the publications that have been produced as the result of the research work.

5.2 MPEG-21-based architecture

This first contribution to the definition of a modular architecture is mainly based on the results obtained in the integration work performed in MPEG-21 regarding DID, REL, RDD and DIP. Therefore, the architecture is focused on the MPEG-21 specifications. It mainly shows how contents can be managed using the Digital Item Processing (DIP) approach.

The architecture here exposed has been published in the Second International Workshop on Multimedia Interactive Protocols and Systems (MIPS 2004) [75]. It will be enhanced and extended in the next chapter, were a more general architecture will be presented.

5.2.1 Specification

As MPEG-21 has not yet defined an architecture for a system implementing the functionality needed to process a digital item containing information associated to different parts of the MPEG-21 standard, we propose here the definition of a Multimedia Information Protection And Management System (MIPAMS), the DMAG-MIPAMS.

The system architecture consists in several modules, each of them providing a part of the functionality. The use of a modular approach allows the addition of new modules as needed.

The architecture distinguishes between the final user and the servers needed to provide the underlying infrastructure. In the middle, we have the “Intermediary”. The functionality of this intermediary element could be also located either on the user side, on the server side or on both, depending on the characteristics of the equipment or terminal used by the final user to access to the multimedia content. Its main functionality is to hide the complexity of the system from a final user perspective. This will allow us to describe the architecture in a very general way, taking advantage of the equipment capabilities independence. For instance, if the final user accesses to the system using a personal computer, then the intermediary functionality could easily be located into his side. On the contrary, if the final user accesses the system through a PDA or a mobile phone, the intermediary functionality should be better located on the server side, possibly having a new server acting as the user in front of the different servers and providing the user the desired content. Another important functionality of the intermediary could be to provide access to different systems (with different content servers, license server, etc), hiding this issue to the final user.

Figure 48 shows the basic modules that could be present in the proposed architecture.

Briefly, the functionality of each module represented in the architecture is the following:

- Content server: It provides the content that final users may request. It can be internally decomposed into several modules, for instance, if we want to separate digital items describing resources from the resource itself or if the content is stored in an external system.
- License server: It provides licensing functionality needed to access the content. It includes license creation and license validation.
- Certification server: It certifies the entities present in the system, including other modules and final users. It includes registration, authentication or key delivery.

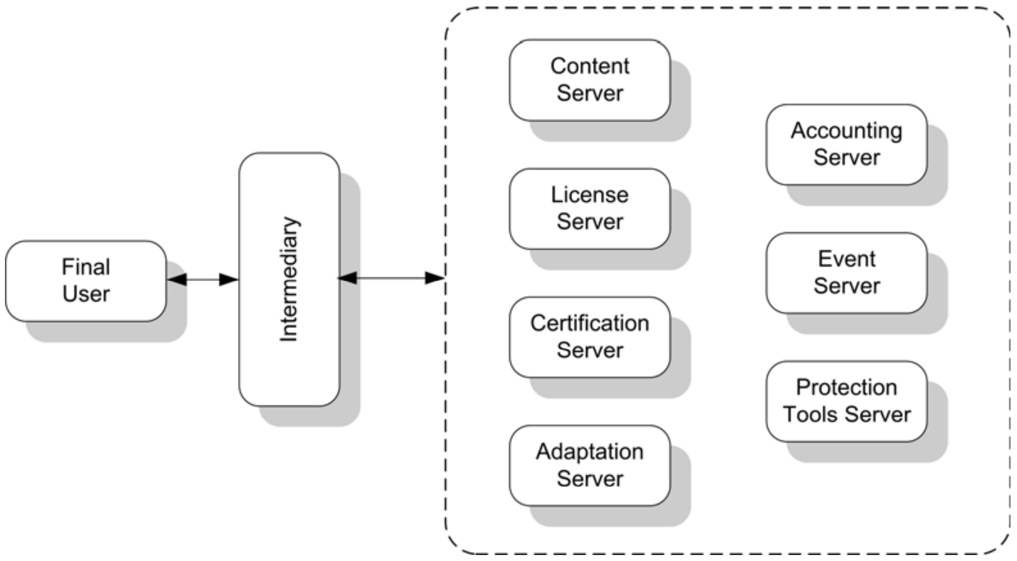


Figure 48. Proposed architecture for the DMAG-MIPAMS.

- Adaptation server: It performs the adaptation of the content depending on the characteristics of the final user terminal.
- Accounting server: Keeps track of what happens in the system, including statistics and traces.
- Event server: Receives events information associated to content usage in order to advise the author or distributor of the content, if needed.
- Protection tools server: It stores the tools needed for the protection of content.

The event server and the adaptation server relate to other two parts of the MPEG-21 standard, Digital Item Adaptation (DIA) and Event Reporting (ER). We do not describe them here, as we want to focus on the rights expression and protection aspects of the system, which will be further developed in next sections.

The protocols appearing in the presented architecture are explained in some more detail in the next section and an example of use is provided in the Use case section.

5.2.2 Protocols involved in the architecture

An important issue in a distributed system is the secure communication between the different entities within it. In the system that we present, it is important to ensure the secure interchange of multimedia content, digital items, licenses and protection or adaptation tools between the different entities, servers and users. These entities communicate using the channel security provided by the SSL/TLS set of protocols. The SSL protocol provides privacy and reliability between two communicating entities. One advantage of SSL is that it is application protocol independent. SSL is a commonly used protocol for managing the security of a message transmission over insecure networks.

In the DMAG-MIPAMS, the transport layer provides a secure authenticated channel, while the application layer provides a message protocol layer that permits the interchange of messages through the secure channel provided by the transport protocol.

The messages exchanged between the different entities in the system have a common structure. The main fields of these messages are an identifier, the content of the message and a digital signature. The content field of the message is defined in a different way depending on the purpose of the message. For example, in messages where the terminal or the intermediary request a license, the information related to the user, content and action is placed in this field. In a message that requests a protection or an adaptation tool, the data that identifies the required tool is placed in this field.

In the use case section, we describe some of the messages interchanged between the Intermediary and the License Server or Protection Tools Server.

5.2.3 Multimedia Content Processing

Currently, the semantics of the standardised DIBOs that form the basis of DIP do not take into account some important concepts in multimedia content distribution such as the protection of multimedia content or the digital rights related to protected or unprotected content.

In this section we present the PlayResource DIBO, which processes protected multimedia content. We also specify its syntax and semantics.

PlayResource DIBO plays the specified protected and governed media resource. As the resource has associated rights expressions, a license-based authorisation is executed in order to check if the user has the appropriate permissions to perform the requested operation. Afterwards, the resource is unprotected with the appropriate IPMP tool, only if the user has been previously authorised. It has as inputs a resource node that represents the DIDL resource element describing the media resource, an MpegDIPResourceChangeObject containing the changes to be applied to the resource before playing it and a Boolean value indicating if the resource will be played asynchronously or not. The return value will be an MpegDIPResourceStatus object identifying the playing resource.

We propose the definition of a set of subroutines that can be used by the implementers of DIBOs that process protected multimedia content that can also have associated rights expressions, such as the presented DIBO or other ones like ExecuteResource, PrintResource or StoreDIDNew. A sample set of proposals for subroutines could be:

- GetLicenses: obtains the licenses related to the operation that a user wants to exercise against a specific resource. It has as inputs the right, resource and the principal. It returns an array of Nodes that contains the REL licenses. The licenses can be retrieved from the License Server or from the DI.
- AuthoriseUser: checks if a user has the permissions to perform the requested operation. It has as input the array of licenses returned by the GetLicenses subroutine. The returned value is a Boolean that is true if the user is authorised or false otherwise.
- GetIPMPInfo: obtains the IPMP information related to the resource against which the user wants to perform the requested operation. This information is obtained from the Digital Item. Relevant IPMP information can be the Tool List that includes the list of IPMP tools used in order to consume the content or the Tool Holder that is where the binary IPMP Tool can be placed if the Digital Item carries the IPMP Tool
- UnprotectContent: obtains the IPMP Tool from the IPMPTools Server and unprotects the multimedia content the user wants to play.

Figure 49 shows how the PlayResource DIBO can use the proposed subroutines when a user wants to play a protected resource.

```
PlayResource(resourceNode, changes, async){
   Node[] licenses=GetLicenses(right,resource,principal);
   boolean isAuth=AuthoriseUser(licenses);
   if(isAuth){
      Node IPMPInfo=GetIPMPInfo(resourceNode);
      if(IPMPInfo != null){
         GetIPMPTools(IPMPInfo);
         UnprotectContent();
      }
      for(x = 0; x < resource.width; x++) {
         for(y = 0; y < resource.height; y++) {
            ShowPixel(x, y, resource[x][y]);
         }
      }
   }
   else
      Alert("Not Authorized");
}
```

Figure 49. PlayResource DIBO.

The DIBO described in this section is the result of a core experiment presented by DMAG in the 69th MPEG meeting held in July 2004 [59]. This core experiment has made use of existing REL tools that we previously developed and contributed as MPEG-21 Reference Software.

5.2.4 Use case

In this section we present a scenario to illustrate how the proposed architecture and protocols and the processing of protected multimedia content are related. Moreover, with this example we try to clarify the relationship among the different MPEG-21 parts presented, i.e. DID, DIP, REL, RDD and IPMP.

The scenario we propose is about a digital library. Imagine a student is subscribed to an electronic journal and wants to view an article. In the architecture we propose, first the user receives an MPEG-21 digital item with the protected multimedia content or a reference to it. Then, when the user accesses the DI, it is parsed and the user can choose the operation or method he wants to perform (view the article, print the article, etc.). Finally, if the user has the appropriate permissions he is authorised to perform the chosen operation.

The workflow of these actions is as specified below and graphically shown in Figure 50:

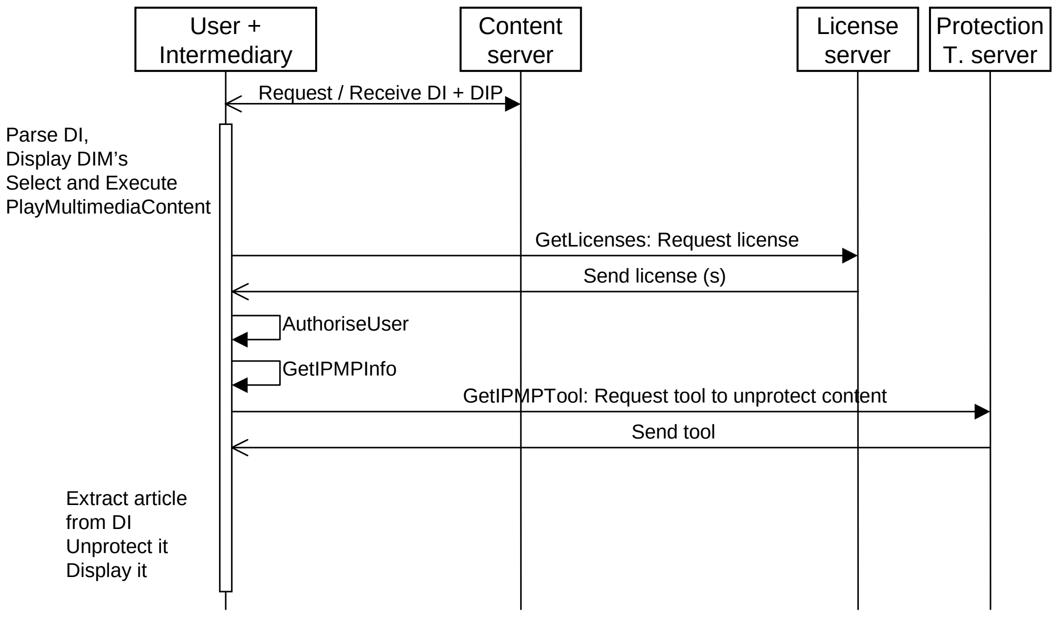

User + Intermediary
Content server
License server
Protection T. server
Request / Receive DI + DIP
Parse DI, Display DIM's Select and Execute PlayMultimediaContent
GetLicenses: Request license
Send license (s)
AuthoriseUser
GetIPMPInfo
GetIPMPTool: Request tool to unprotect content
Send tool
Extract article from DI Unprotect it Display it

Figure 50. Use case workflow diagram.

1. The user receives the Digital Item with the physical resource or a reference to the multimedia Content Server where it is placed. Moreover, the Digital Item has the associated Digital Item Processing information that will allow the user to choose the Digital Item Method that he wants to perform.
2. The user loads the DI in the appropriate application, which displays all the Digital Item Methods.
3. The user chooses a PlayMultimediaContent DIM.
4. PlayMultimediaContent DIM is executed and the PlayProtectedResource DIBO is called. This DIBO performs the following steps:
    a. GetLicenses subroutine calls the Intermediary in order to obtain the licenses associated with the protected multimedia content, the action (play) and the user. Once a secure channel is established between the Intermediary and the License Server, a message is sent to request the license or licenses related to the action. In the content field of the message the user data, the resource and

the right are indicated. Then, the License Server sends a message to the Intermediary and in its content field it places the appropriate REL licenses.

b. AuthoriseUser subroutine performs a license-based authorisation with the REL related information. As the user has a REL license with the appropriate permissions, he is authorised.
c. GetIPMPInfo subroutine obtains the Intellectual Property Management and Protection related information from the Digital Item. It obtains the IPMP Tool information needed to consume the protected resource.
d. GetIPMPTools subroutine calls the Intermediary in order to obtain the IPMP Tool from the Protection Tools Server where it is placed. First, a secure channel is established between the Intermediary and the Protection Tools Server. Then the Intermediary builds a message with the identifier of the IPMP Tool and sends it to the Protection Tools Server. Finally, the requested tool is sent to the Intermediary.
e. The content is obtained from the DI or the Content Server and it is unprotected with the IPMP Tool.
f. The unprotected multimedia content is played.

5.3 MIPAMS Generic architecture

In this section we present the Multimedia Information Protection and Management System (MIPAMS), which is an architecture to manage multimedia information taking into account digital rights management (DRM) and protection. MIPAMS can be seen as an evolution of the architecture presented in section 5.2.

The research work and results presented in this section have been published in:

- IEEE Multimedia [76].
- Third Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS’07) [77].
- Journal of Computer Science and Network Security [78].
- On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops [79].
- First Conference on Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS’05) [80].
- 9th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2005) [81].
- IASTED International Conference Web Technologies, Applications, and Services (WTAS 2005) [82].

5.3.1 Specification

DMAG-MIPAMS is a service-oriented DRM platform and all its modules have been devised to be implemented using the web services approach, which provides flexibility and enables an easy deployment of the modules in a distributed environment, while keeping the functionality independent from the programming language and enabling interoperability.

DMAG-MIPAMS encompasses an important part of the whole content value chain, from content creation and distribution to its consumption by final users.

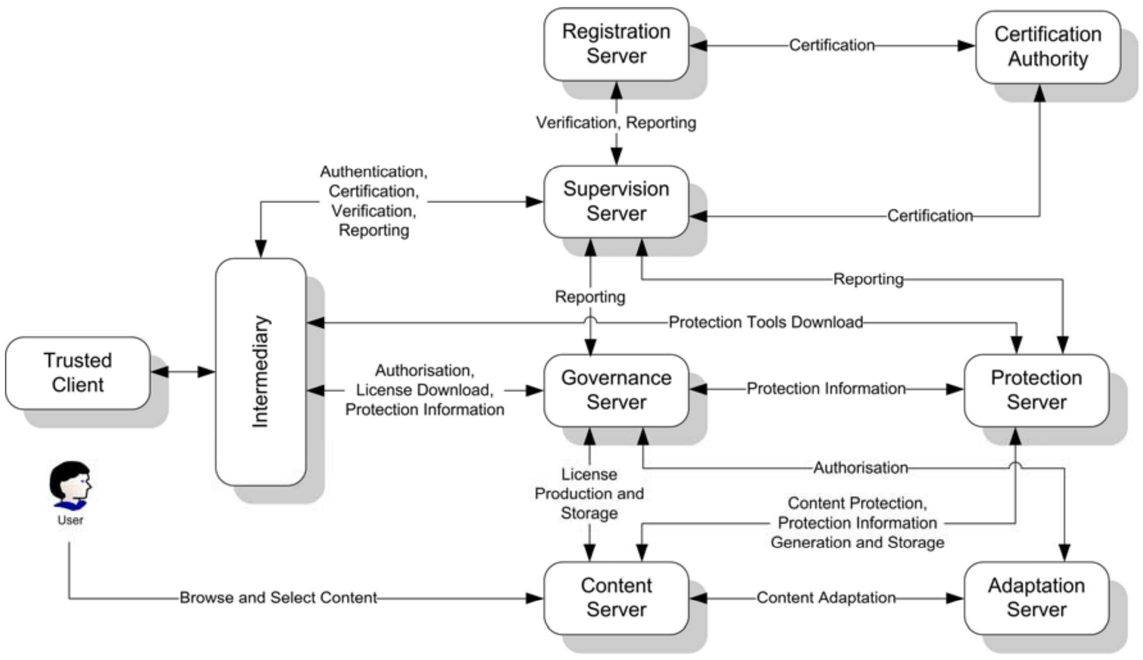


Figure 51. DMAG-MIPAMS architecture.

In the DMAG-MIPAMS architecture context, we can define two important concepts: Components and Actors. An Actor is a user (person or an organisation) that uses a Component. A Component is module or set of software tools that offers a subset of DRM-related functionalities and which interacts with other components.

5.3.1.1 Content Server

This component offers the following functionalities:

- enable users to browse/select content
- provide the content that final users may request to user applications
- encode and add metadata to received raw contents from providers
- register the digital items/objects describing resources (metadata) which can be stored independently from the resource itself (which could be stored e.g. in an external system). This functionality should be available for the Content Server itself as well as for client production tools.

The generation of the encoded and protected objects with metadata is an operation that can be provided as a service in the server part. However, it can be also done by means of specific client tools available for the production environment, which only register the content into the server once it has been protected and the once metadata has been included in the client tool.

5.3.1.2 Adaptation Server

It performs the adaptation of contents and their associated metadata, depending on transmission, storage and consumption constraints. It could be included in the content server as an integral part of it.

The adaptation of metadata can also involve the adaptation of the related licenses, as derived objects or contents can be seen as new creations with regard to original ones.

New licenses must be created during the adaptation process, always respecting the terms and conditions fixed in the original license or licenses for the adapted objects or contents.

5.3.1.3 Protection Server

This module offers the following functionalities:

- protect objects (for content server or client production tools)
- protection tools description and download
- generate protection keys
- store protection keys (optionally here)

The protection of objects is also an operation that can be provided as a service in the server part. However, it can be also done by means of specific client production tools available for the production environment, which protect the objects in the client side.

Protection Server offers a service for protecting the content or digital objects, which become protected objects, mainly using encryption techniques and scrambling. This service can be used by the Content Server. It must create and associate various things to the digital objects. First, the information related to the protection process that has been followed, and, second, the protection techniques, keys or tools that have been used. In this way, it will be possible to determine the reverse process and unprotect the object, if the user is allowed to.

It has also to provide the functionality to download the tools for protection and unprotection in case a user does not have them available in the terminal or device.

When client production tools are used for protecting the content, the client tool is responsible for the protection. However, it can contact the Protection Server to retrieve the list of available protection tools as well as their implementation.

The necessary keys used to protect the content and needed to unprotect it can be stored in this server, or they could be also stored in the governance server together with the corresponding license or even in the content server.

5.3.1.4 Governance Server

The Governance server component includes the following functionalities:

- generate licenses (end-user, distribution, etc.)
- store licenses
- perform online license-based authorisation
- translate licenses

The license generation functionality is only available for content creators, content providers and distributors. For example, a content creator can create a license that grants a content distributor the right to distribute its content under certain conditions. On the other hand, a content distributor will not be authorised to distribute content unless it has the corresponding license issued by a content creator or provider. Final users will not be able to access this functionality unless they act as content distributors (such as in P2P networks) or create their own content for distribution (in this case, they have to be considered as content creators). In any case, any user that creates a license describing the use of some specific content must have the corresponding rights.

The generated licenses are associated to a content or digital object (by means of the object ID), which becomes a governed digital object. Licenses are stored in a license database or repository for authorisation purposes, as we explain next, and delivered to the client when necessary.

The governance server offers authorisation functionality (in an online mode). That is, it is responsible for authorising users to perform actions over resources. Every time a user tries to perform an action over a resource, this module checks in the license database if he has a license that authorises him to do so. As licenses can be written in different languages, as for instance ODRL or MPEG-21 REL, translation functionality is necessary to provide interoperability. In this way, the system can work with a predefined language and convert licenses expressed in other languages to the predefined one under certain conditions. To facilitate this interoperability, we can define "equivalent" profiles between different rights expression languages, to enable their translation in both senses.

5.3.1.5 Certification Server

Certification server involves many different features. It is decomposed into: Certification Authority, Registration server and Supervision server.

Any user must be registered in the system in order to be able to interact with it. Once a user is registered a certificate can be optionally requested to the Certification Authority. The certificate can be used to authenticate him when necessary.

Any tool that interacts with the system must be trusted by the system. One way to ensure this consists in registering the tools in the Registration server prior to its installation in user devices. Later, when a user installs and tries to use a tool for the first time, he will be forced to certify it, which consists in verifying its integrity (Supervision server), registering it in the system as an installed tool together with the device where it has been installed (Supervisor server) and issuing a specific tool certificate (Certification Authority) which can be used to establish secure communications with the server part.

Any tool in the user side must be trusted during its whole life. One way to ensure this is to periodically perform some checks against Supervision server to verify its integrity. Every time a user tries to do an action over a protected and governed object, the client side module can send the necessary information to the server side so that it can verify not only the user status and the device where the tool is being ran, but also the tool integrity to ensure that the module has neither been modified nor corrupted. By ensuring the tool integrity, we can be sure that it is still trusted from the system point of view.

Any module of the system can issue some reports about specific events upon request or in a predefined manner. Supervision server receives these reporting messages that include information about different aspects of media usage. The information it collects is stored in a database. It can be then used by specific applications to keep track of what happens in the system by generating statistics and traces. For example, in a scenario where a content distributor and a client reach the agreement of monthly billing, it can provide the distributor the list of products that the user has consumed for billing purposes. With this information, it can also generate automatic reports to the authorised parties. Supervision server needs to accept online the reports generated during online operation as well as during offline operation. For the latter case, client tools need to provide a mechanism to securely store them locally, and Supervisor server needs to provide a means for checking the list integrity.

Certification Authority

It is the Certification Authority which issues X.509 for the different Components and actors in the system. The main functions of the component are:

- issue X.509 installed tool certificate
- issue X.509 user certificates for registered users
- issue X.509 component certificates for the different architecture servers

Registration Server

It is used to register actors and potentially installable tools. The result of the actor's registration is a user certificate, whereas tools registration is performed to be able to verify them once installed on client devices. The main functions of the component are:

- register tools
- register actors

Supervision Server

It authenticates and supervises actors and system components. Moreover, it is responsible for extracting and registering a fingerprint for installed tools so that they can be verified during their whole life operation and requesting the tool certificate to the Certification Authority. It also verifies the user tool integrity during its operation by checking its fingerprint, registered during certification. Moreover it receives the action reports regarding content consumption or other relevant issues in the system as e.g. license generation. The main functions of the component are:

- authenticate users
- authenticate installed tools
- authenticate architecture components
- verify tool installation attempts against registered tools features
- register new installed client tools and tool and device fingerprints
- request installed tool certificate to the Certification Authority
- receive and store action reports

5.3.1.6 Trusted Client

The trusted client is a module with which the client application must interact to enforce DRM. It consists of a trusted software module and a secure local repository for the storage of licenses, protection information, offline operations reports and other critical data. Main functionalities include:

- estimate trusted client and tool features
- estimate device features
- perform offline authorisation
- unprotect content

- retrieve offline action reports
- store offline action reports
- store content protection information
- install unprotection tools
- license local storage
- tool certificate storage

The intermediary is usually an integral part of the trusted client but could also be located in the server part to simplify the number of Components the trusted client needs to contact. It can be seen as a broker to whom the trusted client requires the authorisation and keys needed to unprotect the content. Main functionalities include:

- require certification to Supervision Server
- require verification to Supervision Server
- require online authorisation to Governance Server
- license download from Governance Server
- send offline operations to Supervision Server
- download unprotection tools from Protection Server

The trusted client could be implemented as a specific client tool or also as a plug-in for an already existing tool. In the latter case, the trusted client is responsible for interacting with the client tool for delivering the unprotected contents to it.

5.3.1.7 Client Application

The client application is the player or edition tool which needs to deal with the protected contents. In the case of the player, it needs to get the unprotected content to be reproduced. In the case of a production tool, it needs to request authorisation before allowing the user act upon the objects as e.g. to modify an image or embed it in another object.

5.3.1.8 Use Case

In this section we present a scenario to illustrate how the proposed architecture and the processing of protected and governed multimedia content are related. It describes content consumption.

Suppose that a user has purchased online a license that grants him the right to play a song during a certain period of time. The acquisition of the license could be performed in various ways. On one hand, the user could have obtained the license in the same place where he purchased the content. In this case, if the license needs to be customised for a particular user, the content distributor must request the license to the corresponding protection and governance servers. On the other hand, the user could have obtained the content through a P2P network, or other online or even offline distribution channels. In this latter case, the content must have some metadata that identifies the content server with which the user must interact to purchase the appropriate license.

The aforementioned user has, installed in their device, a specific tool or plug-in that manages the protected and governed objects of the proposed system and that is able to display them in the appropriate way.

The use case begins when the user downloads a protected and governed song, opens it with their favourite player, which includes the appropriate plug-in and tries to listen to it (Play song).

We will assume that in this case it is not the first usage of the tool in the device, so that the plug-in had been already certified in the system. Although the plug-in has not been manipulated, the system needs to verify its integrity before allowing its operation.

Figure 52 shows the steps involved in the content consumption use case, which are the following:

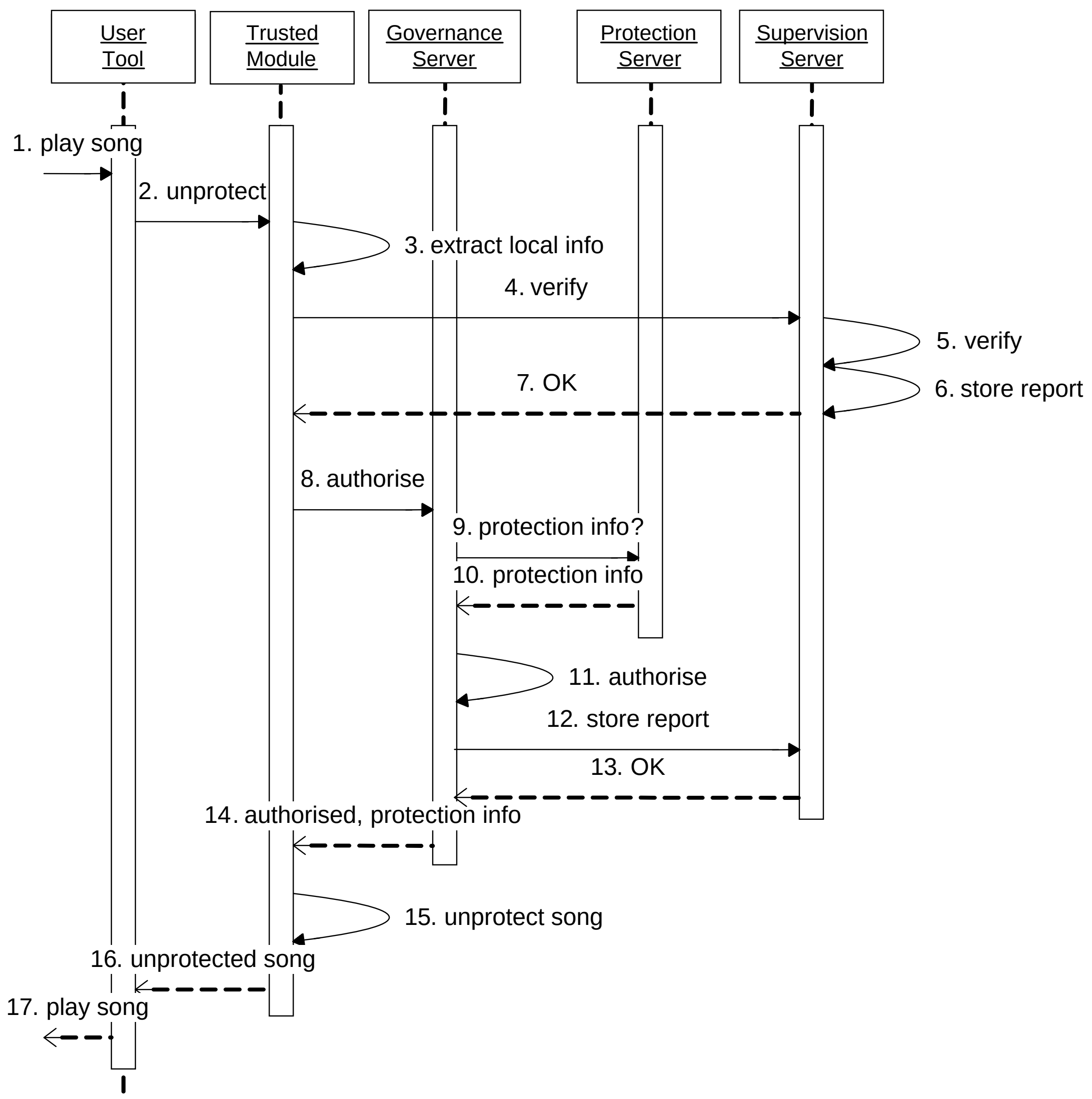


Figure 52. Content consumption Use Case.

1. The user tries to listen to the song in his favourite player.
2. The viewer requires the unprotection of the song to an internal trusted module.
3. The trusted module extracts local information regarding the tool, user and device. It also gets the offline action reports previously performed in the tool, if available.

4. The trusted module contacts Supervision Server and sends the extracted information.
5. Supervision server performs the user and tool authentication and verifies their status and the tool integrity. Supervisor server also checks for the offline action list integrity and stores the performed actions in the corresponding database.
6. Supervision server optionally stores an action report describing the verification results.
7. Supervision server notifies the verification result to the client trusted module.
8. If the verification result is positive, the trusted module requires authorisation to the Governance server.
9. Governance server determines whether the unprotection information is available in Protection server for the requested content.
10. Governance Server retrieves it if available.
11. Governance server performs a license-based authorisation using its license repository.
12. Governance server sends the pertinent action report to Supervision server, denoting the positive or negative authorisation.
13. Supervision server confirms the storage of the report.
14. Governance server notifies the authorisation result to the client trusted module and sends the unprotection information if positive.
15. If the authorisation result is positive, the client trusted module unprotects the song.
16. The unprotected song is sent to the requesting application.
17. The unprotected song is played.

5.3.2 Mapping of other initiatives to MIPAMS

5.3.2.1 MPEG-21

Next, we are going to describe how our architecture has the necessary functionality to cover the different parts of this standard.

DID. Digital objects can have the structure of Digital Items, as defined in the DID part of the MPEG 21 standard.

DIA. Digital Item Adaptation part defines the necessary metadata to describe the significant issues that could be used to satisfy transmission, storage and consumption constraints, as well as Quality of Service management by the various users. The adaptation server defined in our system is responsible for using the DIA information included in the digitals items to produce adapted and customised contents.

REL. Digital Items can include digital licenses written in MPEG-21 REL. Digital Items or objects managed by licenses are called governed digital objects. In our system, the Governance server offers the functionality not only for the creation, validation and edition of licenses, but also for the authorisation of users based on licenses and using the term genealogy defined in the Rights Data Dictionary (RDD, Part 6) [5] of the MPEG-21 standard. RDD comprises a set of clear, consistent, structured, integrated and uniquely identified terms. The structure of the RDD is designed to provide a set of well-defined terms for use in rights expressions.

IPMP. The Intellectual Property Management and Protection part deals with the standardisation of a general solution for the management and protection of Intellectual Property. Digital Items can be protected in order to ensure that the access to the contents is done according to the license terms. The solution lies in the use of digital signatures and encryption techniques over the digital content, which makes it possible to deploy a business model that ensures the accomplishment of the license terms in a controlled way. These kinds of objects are called IPMP DIDL documents that consist of the protected object (or part of the DIDL document) and the IPMP information expressions. IPMP expressions contain protection information, such as the IPMP tools that protect the content, initialisation settings, keys, etc.; and governance information, such as licenses that govern the content or references to these licenses or license services. In our system, the protection server is the responsible for protecting the content and managing the protection keys and tools. It can also generate the protection information required to be included in Digital Items.

ER. Event Reporting is required within the MPEG-21 Multimedia Framework to provide a standardised means for sharing information about Events amongst Peers and Users. Such Events are related to Digital Items and/or Peers that interact with them. In the MPEG-21 context, the reporting messages that include information about different aspects of media usage are called Event Reports. In our system, the management of the event reports is performed by the Supervisor. Whereas the event reports are generated in any of the modules as notification messages, the Supervisor server is in charge of receiving and storing them.

5.3.2.2 OMA DRM

The scope of OMA “Digital Rights Management” [7] is to enable the controlled consumption of digital media objects, to enable superdistribution of DRM Content, and to enable transfer of content between DRM Agents, which are the device-specific elements that enforce the rights associated to the objects.

The OMA DRM system only differentiates between two server functions: content issuers and right issuers. It enables Content Issuers to distribute Protected Content and Rights Issuers to issue Rights Objects (or digital licenses, as we have called them along the paper) for the Protected Content. A Rights Object is cryptographically bound to a specific DRM Agent, so only that DRM Agent can access it.

The mapping of the OMA functionalities into the MIPAMS architecture is summarised in Table 12.

Table 12. OMA vs MIPAMS functionalities.

OMA	MIPAMS
Content issuer	Content Server. Protection Server: content protection, association of protection information, and association of license retrieval information. Adaptation Server.
Rights issuer	Authentication of DRM agents Governance server: all except rights enforcement Protection server: key management

OMA DRM defines a rights expression language (REL) [43] that is based on ODRL [44]. ODRL has a different syntax in comparison with MPEG-21 REL, but the semantics defined by them are quite similar.

Our proposed architecture enables the use of whatever rights expression language the content creators, providers or distributors want to use, associated to digital objects. Internally, our system can work with a predefined rights expression language and provide some translation mechanisms for converting licenses expressed in other languages to the predefined one. This translation can only be performed under certain conditions, which can be grouped to define rights expressions languages profiles, as stated before.

Some extended functionality offered by MIPAMS, and not considered in OMA is described in subsequent sections.

5.3.3 Comparison of MIPAMS with other initiatives

The research work and results presented in this section have been published in:

- IEEE Multimedia [76].
- Journal of Computer Science and Network Security [78].

5.3.3.1 General comparison with OpenSDRM and OMA DRM

OpenSDRM, MIPAMS and OMA-DRM, they all share some commonalities: either the specifications are public, or they provide public documented interfaces, or their source-code is publicly available.

OMA-DRM development has been supported by a wide community of companies and it is widely implemented and deployed in the marked by the major mobile phone suppliers (Nokia, Sony-Ericsson and others). OMA has already launched two versions of its DRM recommendations (the latter reached its maturity in middle 2006) and it is not yet widely implemented in mobile terminals. Both MIPAMS and OpenSDRM were born in the heart of the scientific research community, supported by public financing (National and European), and are mostly used by academic communities. MIPAMS and OpenSDRM continue to evolve with the aid of the community. OpenSDRM has recently been converted to an open-source project, and its source-code can be obtained at Sourceforge.

Although OMA DRM has been developed to address a very specific vertical sector, i.e. the mobile phone industry, it can be used, up to a certain extend, in other different usage scenarios as well. OMA DRM mandates a specific file format (the DCF) to hold the DRM-governed items. This is a key-point in the interoperability of the different devices developed by different manufactures and operated by different mobile Telecom providers. On the other hand, content, rights and platform independence were the three major design goals of OpenSDRM. Therefore, OpenSDRM can be easily adapted to several types of content and to different business models. In fact this has already been done since OpenSDRM has been used in different European research projects, dealing, for example with MPEG-4 and JPEG2000. MIPAMS content format is based on MPEG-21 Digital Items, thus providing enough flexibility to handle many different content formats. Licenses are expressed in MPEG-21 REL, although MIPAMS provides mechanisms for the conversion from and to OMA DRM REL.

Both OMA-DRM and OpenSDRM define DRM architectures that deal directly with the distribution of rights for the final consumer. This means that they have mechanisms (the DRM Agent in OMA and the Wallet in OpenSDRM) that handle all the authorisation clearance and rights parsing at the user-side, having specific protocols for getting the rights from rights issuance entities (Rights Issuer in OMA and the License Server in OpenSDRM). OMA DRM implements a protocol called ROAP (Remote Object Access Protocol), while OpenSDRM uses a specific web-services call entry in the license issuer. MIPAMS extends this model by adding the possibility that the licenses issued are not final user licenses. MIPAMS supports final user as well as distribution licenses. This supposes that a content distributor will need to own a distribution license in order to be able to derive final user licenses from it, something that will be controlled when issuing the end-user license. On the other hand, MIPAMS also supports the usage of license-like documents that can be defined by content creators or content owners, which are useful to state the rights that a distributor could exploit over the content and the related conditions. Any distribution license, when created, must fit the rights and conditions stated in those license-like documents. The authorisation algorithm performed for final users against end-user licenses can optionally involve the verification of the distribution license from which they derive. In this way, the whole content value chain can be controlled.

From an architectural point of view, both MIPAMS and OpenSDRM platform are richer than OMA. From a functional point of view, there are functionalities that are shared between all of the three platforms. However, some extended functionality offered by MIPAMS and not considered in OMA is the following:

- Rights enforcement: MIPAMS enables local or remote enforcement of rights, whereas OMA performs it at the point of consumption.
- Supervision: MIPAMS Supervision Server enables post-usage payment and provides statistical, tracking and control functionalities, which are not considered in OMA.
- Certification and trust on clients: In OMA, the trust on DRM Agents is based on the possession of digital certificates. However, the periodic verification of the client modules integrity is not considered.

In the same direction, OpenSDRM is quite comparable with the MIPAMS DRM platform. In fact, OpenSDRM offers also some extensions when compared to the OMA-DRM platform, such as:

- The license template mechanism present in OpenSDRM allows an easy extension of the Rights Expression Language (REL) supported and OpenSDRM is therefore not particularly tied to any specific REL;
- The Wallet, which has a similar behaviour as OMA’s DRM Agent, is capable of not only handling with registration and authentication processes, license download, enforcement and authorisation, but also with the download of new protection tools;
- OpenSDRM offers the connection to payment functionalities, whereas in OMA no payment functionalities are mentioned;
- OpenSDRM DRM mechanisms are independent of the type of governed content, while OMA specifies its own OMA DCF format.

OMA also presents some advantages over the OpenSDRM and MIPAMS platforms:

- It is widely tested, supported and implemented in most mobile phones (in particular OMA-DRM version 1);
- It uses a common format, which in part simplifies the overall DRM mechanisms.
- A key aspect of DRM is security. In this particular aspect, MIPAMS, OpenSDRM and OMA-DRM provide a secure environment for the DRM operations, making extensive usage of symmetric and asymmetric cryptography, digital credentials and secure protocols.

A key aspect of DRM is security. In this particular aspect, MIPAMS, OpenSDRM and OMA-DRM provide a secure environment for the DRM operations, making extensive usage of symmetric and asymmetric cryptography, digital credentials and secure protocols. Such mechanisms are detailed on the platforms specifications and are out of the scope of this section.

5.3.3.2 Security comparison with OpenSDRM

Regarding component registration and certification, both OpenSDRM and MIPAMS systems enable different mechanisms for the components registration in the system. While OpenSDRM enables a fully automatic registration and certification procedures, the MIPAMS platform uses a non-automatic registration but an automatic certification once the components are validated. This point is not a difficult aspect to be overcome, as the result of the process ends to be the same: a X.509 certificate for the component, although in OpenSDRM X.509 certificates are used to certify DRM components, while the different DRM components functionalities are certified by another different certificate [65]. In this way, independently of the registration and certification processes, a component will own a X.509 digital certificate in both systems. We just need to issue compatible certificates for having compatible components at this level.

Regarding component mutual authentication processes, both systems perform a client-server mutual authentication based on their component digital certificates. On one hand, both systems include a list of trustworthy Certification Authorities in their components. To ensure that the components of both systems will be trusted, we just need to be sure that they are issued by a common Certification Authority or that the CA certificates used in both systems are included in all components. On the other hand, OpenSDRM enables any component to query the Authentication Server for retrieving the component credentials revocation status, whereas MIPAMS does not, assuming that the server components will be controlled. In order for MIPAMS components to be authenticated in OpenSDRM they would need to be registered in the Authentication Server. Something different happens with client components in MIPAMS. Client tools, as we have already explained, are certified and registered in the Supervisor component, and authenticated in the same way of Actors, by using their unique identifier.

OpenSDRM partly uses the same authentication process for Actor and Component functions authentication, querying the AUS to check for the revocation status of their credentials. However, in the specific case of Actor authentication, there is software called Wallet broker that is responsible for handling the Actor authentication processes. In this sense, MIPAMS acts in a different manner. MIPAMS Supervisor authenticates the user by his identifier, which is extracted in the client application and sent in the SOAP messages over the secured channel. In this authentication mechanism, OpenSDRM always recurs to the full credentials, which are sent to ensure a strong authentication process.

In terms of credentials format, there is also some differences between OpenSDRM and the MIPAMS platform. OpenSDRM uses both X.509 certificates for DRM components certification and authentication and a XML mapping of X.509 user certificates for DRM functions and Actors certification and authentication, whereas MIPAMS uses only X.509 certificates. The differences in client authentication can be overcome by: 1) Extracting the user identifier of OpenSDRM XML certificates for authenticating users in MIPAMS; 2) Perform an authentication based on the user identifier instead of the whole XML certificate in OpenSDRM for MIPAMS clients; 3) Using an alternative authentication process based on SAML tokens in order to avoid multiple authentications for the same user, based on digital signatures.

There is a strong difference between both platforms in terms of security design. While in OpenSDRM two security layers coexist to ensure both transport-level and application-level security, MIPAMS depends only on one transport-level security. At the application-level, in the MIPAMS case, the different components assume that there is a secure channel established and authentication processes are somehow shortcut.

5.3.4 Usage of MIPAMS in VISNET II Network of Excellence

5.3.4.1 Introduction

VISNET II builds on the success and achievements of the VISNET Network of Excellence (NoE) to continue the progress towards achieving the NoE mission of creating a sustainable world force in Networked Audiovisual (AV) Media Technologies. VISNET II is a network of excellence with a clear vision for integration, research and dissemination plans. The research activities within VISNET II cover 3 major thematic areas related to networked 2D/3D AV systems and home platforms. These are:

- Video Coding
- Audiovisual Media Processing
- Security

VISNET II brings together 12 leading European organisations in the field of Networked Audiovisual Media Technologies. The consortium consists of organisations known for their proven track record as well as both national and international reputations in audiovisual information technologies. VISNET II integrates a number of researchers who have made significant contributions to the advance of this field of technology through standardisation activities, international publications, conferences and workshops activities, patents as well as many other prestigious achievements. The 12 integrated organisations represent 7 European states spanning across a major part of Europe, thereby promising the efficient dissemination of resulting technological development and exploitation to larger communities.

Video Surveillance and Virtual collaboration are the two application domains chosen in VISNET II NoE to integrate the research results and software modules developed in the video coding, processing and security theme areas.

Next section presents the work done regarding the definition of a virtual collaboration scenario, in which three organisations have set up a collaborative working environment using the MIPAMS DRM architecture. According to this scenario, some relevant use cases have been defined.

5.3.4.2 Virtual Collaboration scenario

This section presents a set of relevant uses cases defined for the virtual collaboration scenario in which three organisations, Aa, Bb and Cc, have set up a collaborative working environment. In this context, the MIPAMS DRM architecture allows them to work together on a project to design and produce a new widget.

This collaborative working environment includes a shared data repository that stores protected data, which can be downloaded by any user, but which can only be used according to the license terms. In this scenario we can distinguish three different phases: widget design, review and production. During the design phase a single document is used to define the specifications for the widget. Designers in Aa and Bb will jointly work in the document for five weeks. At the end of this period, the document will be reviewed by the project managers from the three organisations. After the review, the document will be released to allow engineers from Bb and Cc to start working on the production of a prototype widget. As the production of the widget is considered highly commercial sensitive by the three organisations, it has been decided that at all stages of the process, the access and usage rights will be restricted.

A walk-through on how the DRM architecture is used in this scenario is given below. Moreover, a use case has been defined for each of the different phases defined in the collaborative working scenario.

Design phase use cases

In this section we present two use cases to illustrate how the access to the edition of the widget design document is managed by the MIPAMS DRM architecture for the different users in the system. It is assumed that a small number of employees are assigned to the designer role in organisations Aa and Bb. Only these employees are able to work on the joint design document.

Then, for the design phase, two different use cases have been defined. In the first use case a designer from Aa tries to modify the specification document. In this case, as they have the appropriate permissions, they are able to load and modify it. In the second use case, an engineer from Cc tries to edit the widget design document. In this case, as they do not have the appropriate permissions, they are not able to access the document.

In the first use case, Alice, who is a designer in organisation Aa, wants to modify the current design document. In this use case we assume that Alice was registered in the system as a designer in organisation Aa so that she has a license that proves her role. The widget design document has already been created and it is governed by a license that only grants designers belonging to organisations Aa and Bb the right to modify the protected document.

The use case begins when Alice downloads the protected and governed design document, opens it with a text processor, which includes the appropriate plug-in to manage protected content, and tries to load the widget design document.

Figure 53 shows the steps involved in the management of the design document use case, which are the following:

1. Alice authenticates herself in the system, using her username and password and receives and authentication token that is used to authenticate the user when interacting with other services.
2. Alice tries to load the widget design document in her editor.
3. The viewer requires the unprotection of the document to an internal trusted module.
4. The trusted module requires authorisation to the Governance server.
5. Governance server determines whether the unprotection information is available in Protection server for the requested content.
6. Governance Server retrieves it if available.
7. Governance server performs a license-based authorisation using its license repository. In this use case the authorisation result is positive, as Alice is a designer from organisation Aa. Thus, she has the appropriate permissions to load and modify the widget design document.

8. Governance server sends the pertinent action report to Supervision server, denoting the positive authorisation.
9. Supervision server confirms the storage of the report.
10. Governance server notifies the positive authorisation result to the client trusted module and sends the unprotection information, as in this use case the authorisation result has been positive.
11. The client trusted module unprotects the document.
12. The unprotected document is sent to the requesting application.
13. The unprotected document is loaded.
14. Alice makes some changes in the design document and attempts to save the new version of the document.
15. The trusted module requires authorisation to the Governance server.
16. The authorisation is positive, since Alice is designer in organisation Aa and she can modify the document.
17. Governance server sends the pertinent action report to Supervision server, denoting the positive authorisation.
18. Supervision server confirms the storage of the report.

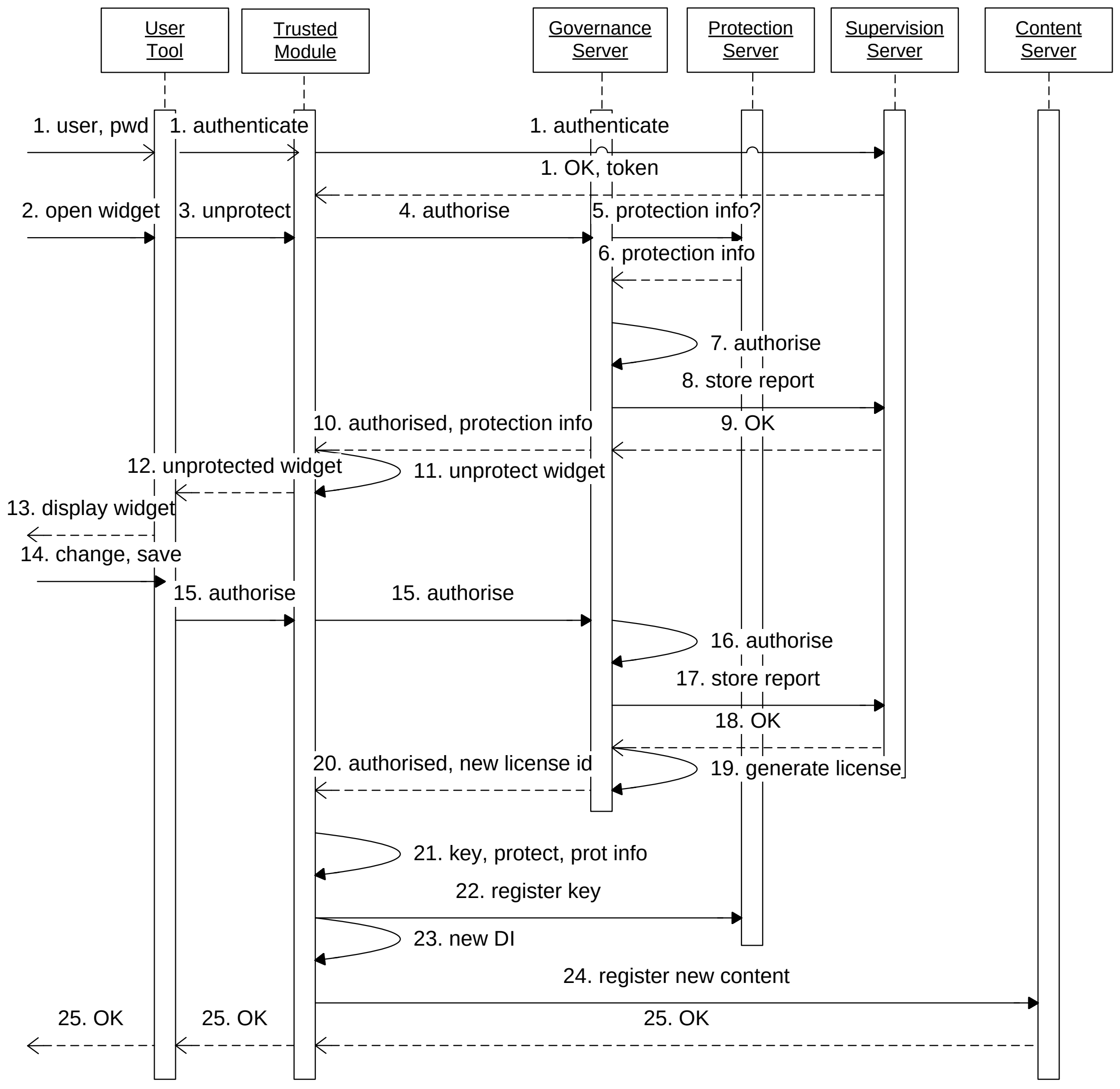


Figure 53. Design phase use case – Positive authorisation.

19. Governance server generates and stores the usage license for the new version of the design document based on the terms present in the license governing the original content.
20. Governance server notifies the positive authorisation result to the client trusted module and returns the identifier of the new license
21. The trusted module uses one of the protection tools it has to protect the content, generates a protection key for the content and generates the protection information that will be associated to the DI.
22. The trusted module registers the protection key in the Protection server
23. The client trusted module creates a new DI. It contains the protected version of the document, a reference to the license governing the protected content, the protection information and the processing information.
24. The DI is stored in the Content Server.
25. Confirmation is sent to the user

In the second use case Peter, who is an engineer in organisation Cc, wants to modify the current design document. In this use case we assume that Peter was registered in the system as an engineer of organisation Cc so that he has a license that proves his role. The widget design document has already been created and it is governed by a license that only grants designers of organisations Aa and Bb the right to modify the protected document.

The use case begins when Peter downloads the protected and governed design document, opens it with a text processor, which includes the appropriate plug-in to manage protected content, and tries to load the widget design document.

Figure 54 shows the steps involved in the management of the design document use case, which are the following:

1. Peter authenticates himself in the system, using his username and password and receives and authentication token that is used to authenticate the user when interacting with other services.

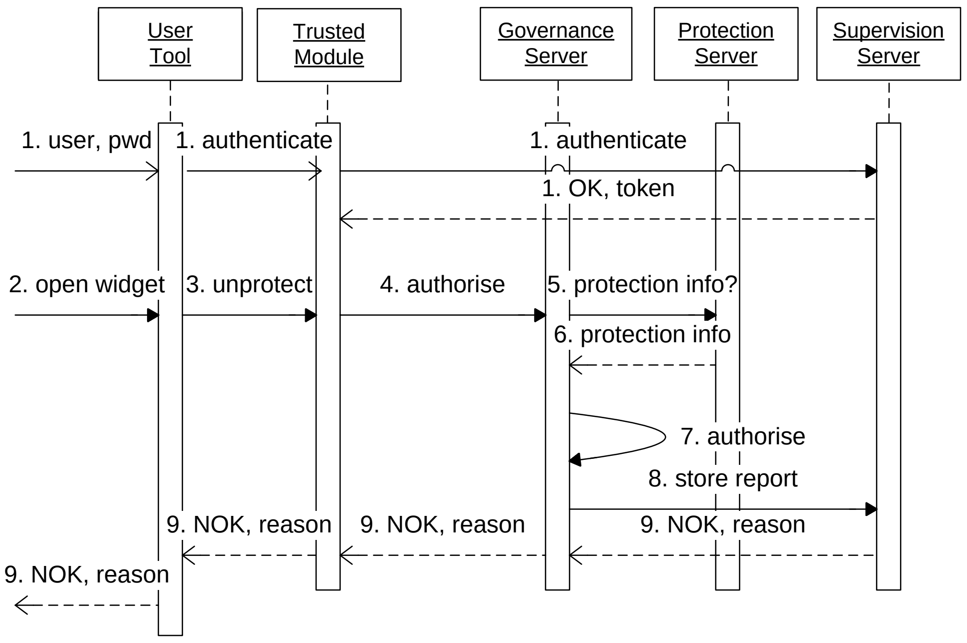


Figure 54. Design phase use case – Negative authorisation.

2. Peter tries to load the widget design document in his player.
3. The viewer requires the unprotection of the document to an internal trusted module.
4. The trusted module requires authorisation to the Governance server.
5. Governance server determines whether the unprotection information is available in Protection server for the requested content.
6. Governance Server retrieves it if available.
7. Governance server performs a license-based authorisation using its license repository. In this use case the authorisation result is negative, since Peter is an engineer from organisation Cc. Thus, he does not have the appropriate permissions to load and modify the widget design document.
8. Governance server sends the pertinent action report to Supervision server, denoting the negative authorisation.
9. An informative message is shown to Peter pointing out the reasons why he cannot modify the document.

Review phase use cases

In this section we present two use cases to illustrate how the document is managed by the collaborative working environment during the review phase. During this phase, the completed design document can only be reviewed by the Project Managers. It is assumed that each organisation has one nominated Project Manager, who is able to read the widget design document during the review phase and review the document.

For the review phase, two different use cases have been defined. In the first use case, the Project Manager from organisation Bb tries to review the design document. As they have the appropriate permissions, they are allowed to view and modify it. In the second use case, an engineer from Aa tries to view the widget design document. As they do not have the appropriate permissions, they do not have access to the document.

In the first use case, Charlie, who is the Project Manager in organisation Bb, wants to review the current design document. In this use case we assume that Charlie is registered in the system as a Project Manager of organisation Bb so that he has a license that proves his role. On the other hand, the widget design document was created during the design phase and it is governed by a license that only allows project managers of organisations Aa, Bb and Cc to review the protected document.

The use case begins when Charlie downloads the protected and governed design document, opens it with his text processor, which includes the appropriate plug-in to manage protected content, and tries to load it.

Figure 55 shows the steps involved in the review of the document, which are the following:

1. Charlie authenticates himself in the system, using his username and password and receives and authentication token that is used to authenticate the user when interacting with other services.
2. Charlie tries to load the widget design document in his player.
3. The viewer requires the unprotection of the document to an internal trusted module.
4. The trusted module requires authorisation to the Governance server.
5. Governance server determines whether the unprotection information is available in Protection server for the requested content.

6. Governance server retrieves it if available.
7. Governance server performs a license-based authorisation using its license repository. In this use case the authorisation result is positive, as Charlie is a Project Manager from organisation Bb. Thus, he has the appropriate permissions to load and modify the design document.
8. Governance server sends the pertinent action report to Supervision server, denoting the positive authorisation.
9. Supervision server confirms the storage of the report.

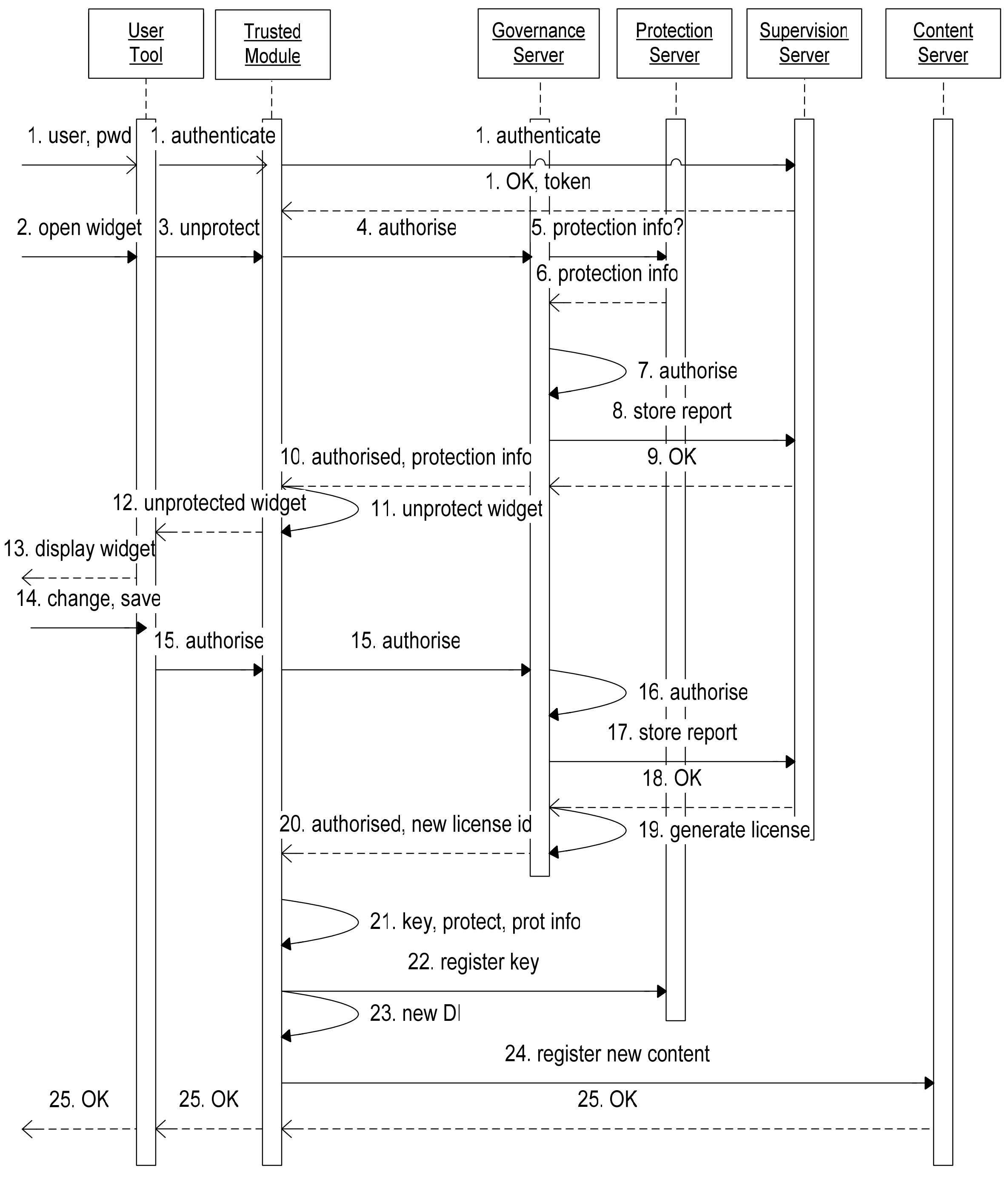


Figure 55. Review phase use case – Positive authorisation.

10. Governance server notifies the positive authorisation result to the client trusted module and sends the unprotection information, as in this use case the authorisation result has been positive.
11. The client trusted module unprotects the document.
12. The unprotected document is sent to the requesting application.
13. The unprotected document is loaded.
14. Charlie reviews the design document and attempts to save the new version of the document.
15. The trusted module requires authorisation to the Governance server.
16. As Charlie is a Project Manager from organisation Bb, he has the appropriate permissions to modify the design document
17. Governance server sends the pertinent action report to Supervision server, denoting the positive authorisation.
18. Supervision server confirms the storage of the report.
19. Governance server generates and stores the usage license for the new version of the design document based on the terms present in the license governing the original content.
20. Governance server notifies the positive authorisation result to the client trusted module and returns the identifier of the new license
21. The trusted module uses one of the protection tools it has to protect the content, generates a protection key for the content and generates the protection information that will be associated to the DI.
22. The trusted module registers the protection key in the Protection server
23. The client trusted module creates a new DI. It contains the protected version of the document, a reference to the license governing the protected content, the protection information and the processing information.
24. The DI is stored in the Content Server.
25. Confirmation is sent to the user

In the second use case Sue, engineer in organisation Aa, wants to view the current design document. In this use case we assume that Sue was registered in the system as an engineer of organisation Aa. Thus, she has a license that proves her role. On the other hand, the widget design document was created during the design phase and it is governed by a license that only allows project managers of the three organisations to modify the document.

The use case begins when Sue downloads the protected and governed design document, opens it with her text processor, which includes the appropriate plug-in to manage protected content, and tries to load the widget design document.

Figure 56 shows the steps involved in the management of the design document use case, which are the following:

1. Sue authenticates herself in the system, using her username and password and receives and authentication token that is used to authenticate the user when interacting with other services.
2. Sue tries to load the widget design document in her player.
3. The viewer requires the unprotection of the document to an internal trusted module.
4. The trusted module requires authorisation to the Governance server.
5. Governance server determines whether the unprotection information is available in Protection server for the requested content.

6. Governance Server retrieves it if available.
7. Governance server performs a license-based authorisation using its license repository. In this use case the authorisation result is negative, since Sue is an engineer from organisation Aa. Thus, she does not have the appropriate permissions to review the widget design document.
8. Governance server sends the pertinent action report to Supervision server, denoting the negative authorisation.
9. An informative message is shown to Sue pointing out the reasons why she cannot access the document.

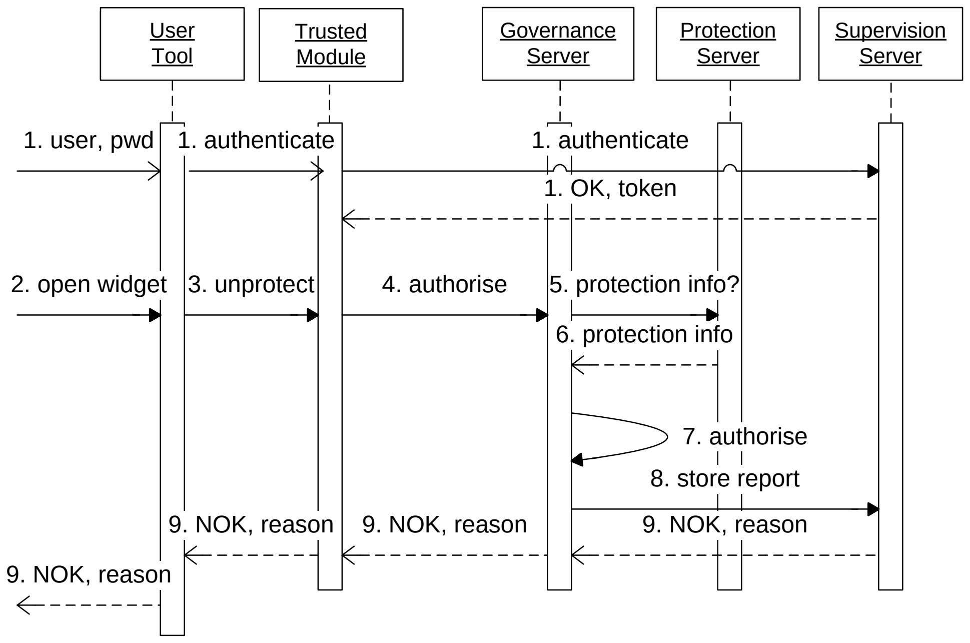


Figure 56. Review phase use case – Negative authorisation.

Production phase use cases

In this section we present two use cases to illustrate how the widget design document is managed in the collaborative working environment by the different users of the system during the production phase. During this last phase, the engineers of the three organisations can access the document, but they cannot make changes to it.

Two different use cases have been defined for the production phase. In the first use case an engineer from Aa tries to view the specifications document and, as they have the appropriate permissions, the document is loaded. On the other hand, in the second use case an engineer from Cc tries to modify the design document and, and they do not have the appropriate permissions, they are not able to save the modification that they have done in the design document.

In the first use case, Lora, who is engineer in organisation Aa, wants to view the final version of the design document. In this use case we assume that Lora was registered in the system as an engineer of organisation Aa. Thus, she owns a license that proves her role. On the other hand, the final version for the design document has been created and it is governed by a license that allows project managers, designers and engineers of the three organisations to view the document.

The use case begins when Lora downloads the protected and governed design document, opens it with her viewer, which includes the appropriate plug-in to manage protected content, and tries to load the widget design document.

Figure 57 shows the steps involved in the view of the design document use case, which are the following:

1. Lora authenticates herself in the system, using her username and password and receives and authentication token that is used to authenticate the user when interacting with other services.
2. Lora tries to load the widget design document in her editor or player.
3. The viewer requires the unprotection of the document to an internal trusted module.
4. The trusted module requires authorisation to the Governance server.
5. Governance server determines whether the unprotection information is available in Protection server for the requested content.
6. Governance Server retrieves it if available.
7. Governance server performs a license-based authorisation using its license repository. In this use case the authorisation result is positive, as Lora is an engineer from organisation Aa. Thus, she has the appropriate permissions to load and modify the widget design document.
8. Governance server sends the pertinent action report to Supervision server, denoting the positive authorisation.
9. Supervision server confirms the storage of the report.
10. Governance server notifies the positive authorisation result to the client trusted module and sends the unprotection information, as in this use case the authorisation result has been positive.
11. The client trusted module unprotects the document.
12. The unprotected document is sent to the requesting application.
13. The unprotected document is loaded.

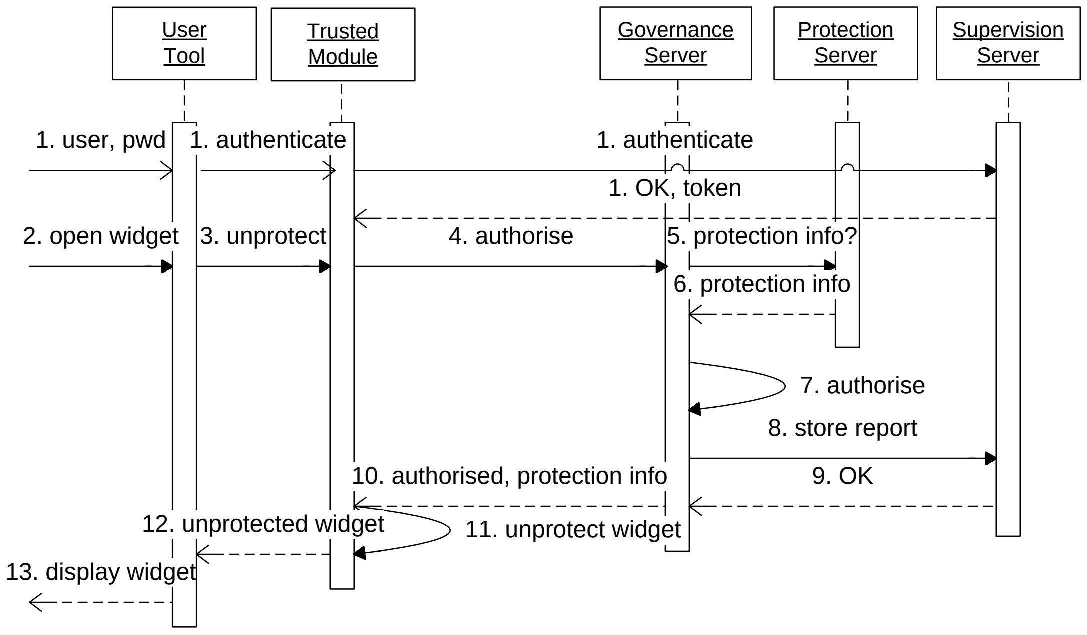


Figure 57. Production phase use case – Positive authorisation.

In the second use case Bob, who is an engineer in organisation Cc, wants to view and modify the final version of the design document. In this use case we assume that Bob was registered in the system as an engineer of organisation Cc. Thus, he has a license that proves his role. On the other hand, the final version for the design document was created during the review phase and it is governed by a license that allows project managers, designers and engineers of the three organisations to view the document.

The use case begins when Bob downloads the protected and governed design document, opens it with his player, which includes the appropriate plug-in to manage protected content, and tries to load the widget design document.

Figure 58 shows the steps involved in the edition of the design document use case during the production phase, which are the following:

1. Bob authenticates himself in the system, using his username and password and receives and authentication token that is used to authenticate the user when interacting with other services.
2. Bob tries to load the widget design document in his player.
3. The viewer requires the unprotection of the document to an internal trusted module.
4. The trusted module requires authorisation to the Governance server.

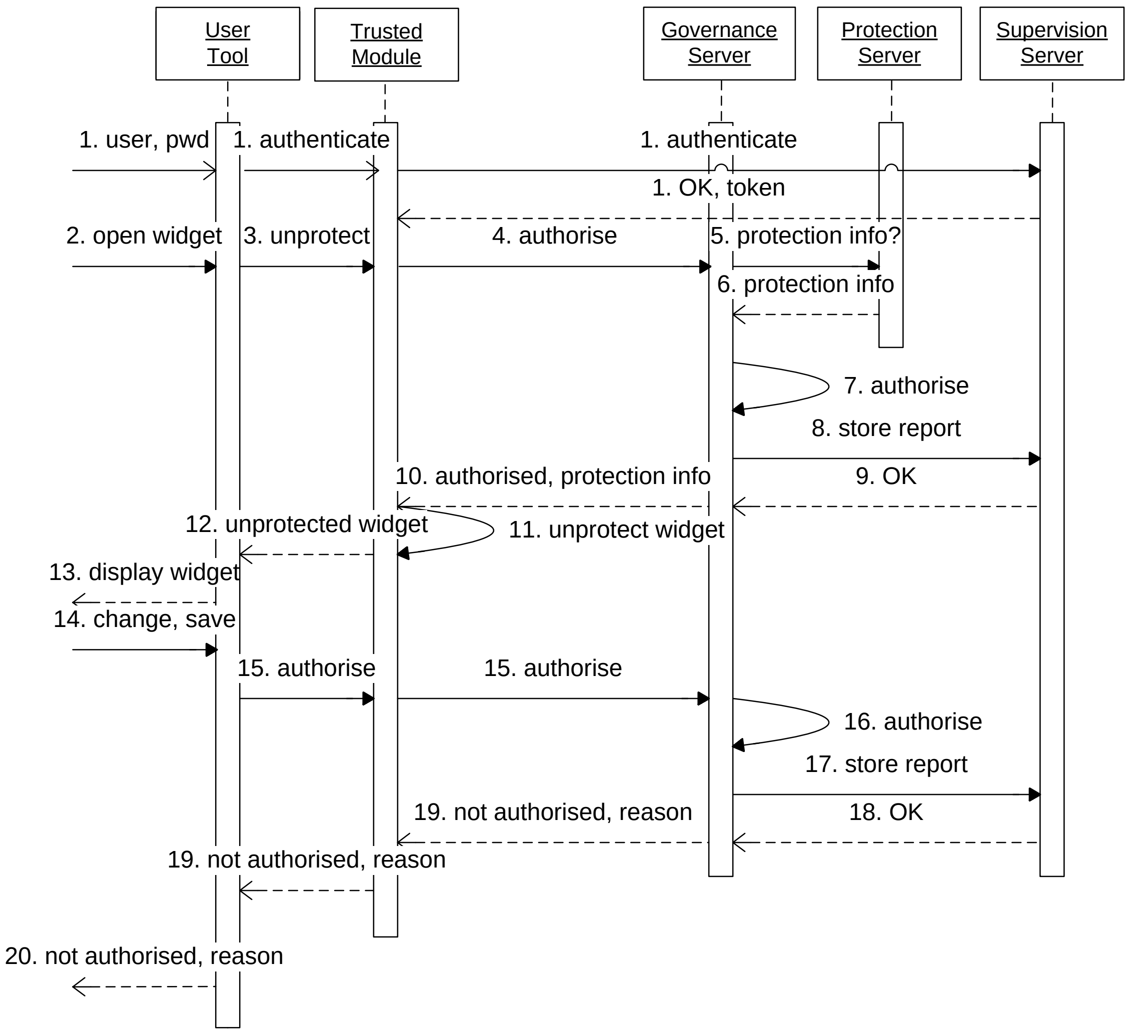


Figure 58. Production phase use case – Negative authorisation.

5. Governance server determines whether the unprotection information is available in Protection server for the requested content.
6. Governance Server retrieves it if available.
7. Governance server performs a license-based authorisation using its license repository. In this use case the authorisation result is positive, as Bob is an engineer from organisation Cc. Thus, he has the appropriate permissions to view the design document.
8. Governance server sends the pertinent action report to Supervision server, denoting the positive authorisation.
9. Supervision server confirms the storage of the report.
10. Governance server notifies the positive authorisation result to the client trusted module and sends the unprotection information, as in this use case the authorisation result has been positive.
11. The client trusted module unprotects the document.
12. The unprotected document is sent to the requesting application.
13. The unprotected document is loaded.
14. Bob reviews the design document and attempts to save the new version of the document.
15. The trusted module requires authorisation to the Governance server.
16. Since Charlie is an engineer, he does not have permissions to modify the design document.
17. Governance server sends the pertinent action report to Supervision server, denoting the negative authorisation.
18. Supervision server confirms the storage of the report.
19. Governance server notifies the negative authorisation result to the client trusted module.
20. The trusted module shows an informative message pointing out the reasons why Bob cannot save the modification he has done in the document.

5.3.5 Usage of MIPAMS in GILDDA

5.3.5.1 Introduction

The goal of GILDDA Project is to develop a web portal that implements the functionality of a bookseller. This portal will be able to distribute any book from the publishers that become associated to the service.

The edition of the books will be performed in a local print works company, as close as possible to the final client. The delivery of the printed product will be done directly to the end user if they collect them in the prints works or otherwise by means of a messenger company that will ship it to the user’s home.

In this way, the prices can be reduced, as currently about the 50% of the final price are perceived by the intermediaries in the distribution chain.

5.3.5.2 Use cases

Next, we describe four uses cases defined in GILDDA Project, which take into account the needs of publishers.

Use Case 1: Selling books online

1. A user accesses a web portal, which acts as a distributor of several publishers.
2. The user browses the books catalogue and selects a book.

3. The user buys a book copy and selects the print works company where they will pick up the book printed hard copy or the messenger company that will ship it to their home.
4. The selected print works company receives a license that grants permission to print a copy of the book together with the protected book.
5. The print works company requests authorisation to print the book, retrieves the protection information, decrypts the book, produces the copy and delivers it to the user.

Use Case 2: Selling books online (Watermarking)

1. A user accesses a web portal, which acts as a distributor of several publishers.
2. The user browses the books catalogue and selects a book.
3. The user buys a book copy and selects the print works company where they will pick up the book printed hard copy or the messenger company that will ship it to their home.
4. The selected print works company receives a license that grants permission to print a copy of the book together with the book.
5. The print works company produces the book copy, which contains a visible watermark regarding the user identification, and delivers it to the user.

Use Case 3: Selling books online from a print works company or book shop

1. A user accesses a web portal, which acts as a distributor of several publishers. The access is done from a host in a print works company or book shop point of sale.
2. The user browses the books catalogue and selects a book.
3. The user buys a license that grants them to print a hard copy of the selected book
4. The user prints a hard copy of the book in one of the printers of the print works company or book shop.

Use case 4: Document rendering

1. A user accesses a web portal, which acts as a distributor of several publishers.
2. The user browses the books catalogue and selects a book.
3. The user buys a license that grants them to view the book.

Next, we describe more in detail the second use case, which is similar to use case 1, but more complete, as it includes the insertion of watermarks in the document.

The use case begins when a user accesses a web portal, which acts as a distributor of several publishers. We assume the user has been already registered in the system.

1. A user accesses the web portal and provides their name and password, which are used to get an authentication token that will be used to authenticate to other services.
2. The user browses the catalogue, selects a book, and views the selling conditions.
3. The user decides to buy a hard copy of the book and selects the print works company that will produce and deliver them the copy.
4. The request is made to the Content service.
5. The Content service requests the protection of the content to the Protection service.

6. The Protection service inserts a watermark into the content and encrypts the content.
7. The Protection service returns the encrypted content.
8. The Content service generates a digital object which contains the digital book and related metadata (event report request, protection information).

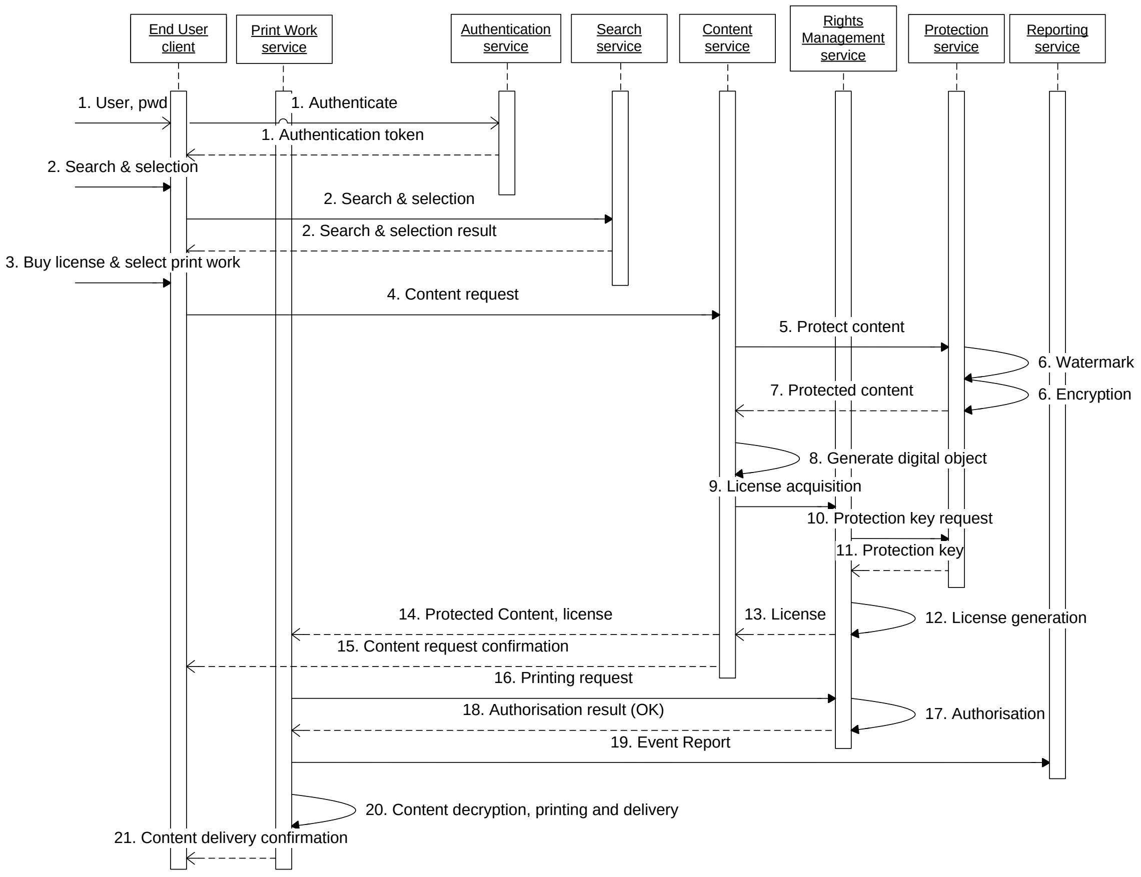


Figure 59. Selling watermarked books Use Case.

9. The Content service requests a license for the user.
10. The Rights Management service requests the protection key for the content to the Protection service.
11. The Protection service returns the protection key.
12. The Rights Management service generates a user license for the print works company, as requested by the user. The license contains the rights and conditions and the protection key.
13. The license is returned to the Content service.
14. The Content service sends the protected content to the print works company service together with the encrypted license.
15. The operation is confirmed to the end user.
16. The print works company tries to produce the book hard copy. An authorisation request is generated.
17. The Rights Management service performs the authorisation.
18. The authorisation result is returned.
19. An event report is generated towards the Reporting service.
20. The document is decrypted and the hard copy of the book is printed. The hard copy contains a visible watermark including the end user identification.

21. The print works company confirms the production and delivery of the book copy.

5.3.5.3 Architecture

This section presents the GILDDA architecture, which enables the management and distribution of digital books, taking into account the copyright. The following architecture is based on MIPAMS architecture, preserving the service-oriented approach.

An additional service has been added with respect to the MIPAMS architecture. It is that of the print works company, which is capable of receiving printing requests, as depicted is previous use case.

Figure 60 depicts the architecture integral modules, which are summarised next:

Content service. This service generates digital objects that consist of the digital content (e.g. digital book or chapter) plus the related metadada (e.g. author, title, publisher). The objects are protected so that only the authorised users have access to it.

User registration service. This service enables the registration of end users and publishers in the system.

Authentication service. This service is responsible for giving authentication credentials to the users in the system. It is based on SAML token authentication.

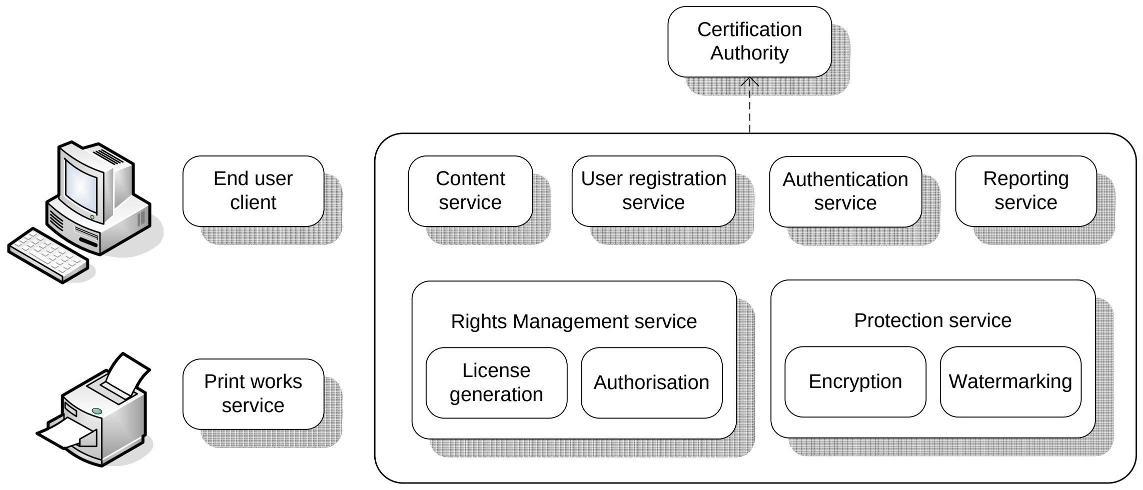


Figure 60. GILDDA Architecture.

Event reporting service. This service receives and stores reports about content usage in the system. The reports enclose some information related to the event such as the action that has occurred, the user that has performed the action, the time when the action occurred, etc. This information can be used for billing purposes.

Content protection service. This service offers the mechanisms to protect digital objects or the content included in the objects. It also generates the protection information (e.g. MPEG-21 IPMP) and inserts it into the protected objects. On the other hand, this service deals with the generation and insertion of watermarks into digital resources.

Rights management service. This service offers two different functionalities:

- License generation: it issues not only distribution licenses but also end user consumption licenses.
- Authorisation: it determines whether a user is authorised to perform the requested operation, according to the terms included in the user license or licenses.

Search service. This service enables the user to search and browse amongst the contents in the catalogue, according to different searching criteria.

End user client. The end user client is a specific tool or placer that is capable of processing the protected and license-governed digital objects. The client asks for authorisation, decrypts the content when the user is authorised to and renders the resource so that it can be viewed, depending on the user rights.

Print works service. This service is capable of receiving requests for printing the resources embedded in digital objects. This service asks for authorisation, decrypts the content when the user is authorised to and renders the resource so that it can be printed, depending on the user rights.

5.3.6 Usage of MIPAMS in AXMEDIS

The Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS) Integrated Project, is an initiative funded by the European Commission under the Sixth Framework Programme (FP6). The DMAG research group is a partner of the IP and, among other tasks, we have been implementing some tools and modules related to protection, governance and certification to create an AXMEDIS framework.

The AXMEDIS Framework supports and automatises the digital content management along the whole value chain. It offers an enlarged set of tools for content that are capable to satisfy even the most complex requests of business. With AXMEDIS multi-channel distribution the same content can be provided for streaming or downloading for different platforms and channels: Satellite, PC Windows or Mac, PDA, Mobile, Set-Top-Box, IP-TV, and many others.

AXMEDIS tools can be used for:

- Multi-channel production and distribution for broadcasting, IP/Internet, WEB sites, P2P, mobile, PDA, IPTV, interactive TV and channels. By using the AXMEDIS Content Processing solution (AXCP) it is possible to manage automatically: content, metadata and licensing information with the operations of ingestion, crawling, database management, indexing, processing, adaptation, transcoding, encoding, decoding, descriptor extractor, recognition, filtering, production, archiving, storing, packaging, preview, extracting fingerprint, licensing, AXMEDIS DRM, profiling, protection, encryption, accounting, enrichment, network management, etc.
- Controlled P2P network for content sharing and distribution, involving customers and business: Content owners and distributors can to exploit the capabilities of P2P protocols to create efficient, controllable, legal and secure P2P networks for content distribution and sharing. By using the AXMEDIS P2P and AXCP solution a distributor may publish content in the P2P network. The content may freely navigate among peers with the supervision and control of the AXMEDIS protection and monitoring tools.
- Exploitation of different business models and/or transactions on the same distribution channels: Automatic content production, protection and distribution with the AXMEDIS DRM solutions will help reduce the cost for content post-

production and management. The AXMEDIS DRM solutions allow to automatically product, protect and distribute content with DRM allowing the reduction of costs for content post-production and management. Content can distributed according to different business models (pay per play, monthly rate, etc.), different rights (play, print, etc.), with different conditions (times of play, duration, etc.).

- Advanced interactivity with cross media models: AXMEDIS Editor and Players support many types of cross-media interactive contents with and without DRM support, from simple multimedia files to complex collections for a large range of applications, from business to business to personal and/or global scale production, protection and distribution, it is possible to create autonomous intelligent dynamic content to be send in the hand of users.

The AXMEDIS multi-channel DRM, an open and interoperable solution for the protection and rights management for a wide range of content, from single files to complex cross media and multimedia, distributed on different channels towards different type of players and devices such as DVB-T, DVB-S, PC, Internet, P2P, PDA, Mobiles, STB.

The AXMEDIS consortium (consisting of leading European digital content producers, integrators, aggregators, and distributors; and also information technology companies and research groups) is to create the AXMEDIS framework to provide innovative methods and tools to speed up and optimise content production and distribution, up to the production-on-demand capability, for leisure, entertainment and digital content valorisation and exploitation in general. AXMEDIS format can include any other digital formats and will exploit and improve other formats such as MPEG-4, MPEG-7, MPEG-21, as well as other de facto standards.

The research work and results presented in this section have been published in:

- Third Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'07) [77].
- On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops [79].
- First Conference on Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS'05) [80].

5.3.6.1 AXMEDIS DRM Architecture

The architecture implemented in the AXMEDIS project consists on several independent modules that interact as web services when they are located in different machines or directly in other situations.

The general description of the AXMEDIS architecture main modules, depicted in Figure 61, is as follows:

- **Protection Processor**. This client tool module is responsible for estimating the client tool fingerprint, enabling or disabling the tool, verifying the tool integrity and unprotecting protected multimedia objects that are then passed to the modules in charge of editing or rendering the multimedia objects inside an AXMEDIS tool.
- **Protection Manager Support Client (PMS Client)**. This client tool module manages and stores protection information, licenses, reports regarding the offline performed actions and other secured information in a local secure storage system called secure cache. It is responsible for authorising users to perform actions over

objects with respect to digital licenses during offline operation. It also delivers protection information to Protection Processor, if present in the secure cache, or requests it to AXCS (introduced later) after a positive authorisation. It acts also as the intermediary module used by Protection Processor to contact AXCS to certify and verify tools.

- **Protection Manager Support Server (PMS Server)**. This server side module is responsible for authorising users to perform actions over objects in an online environment and requesting protection information to AXCS if needed. It acts also as an intermediary module to contact AXCS from PMS Client.

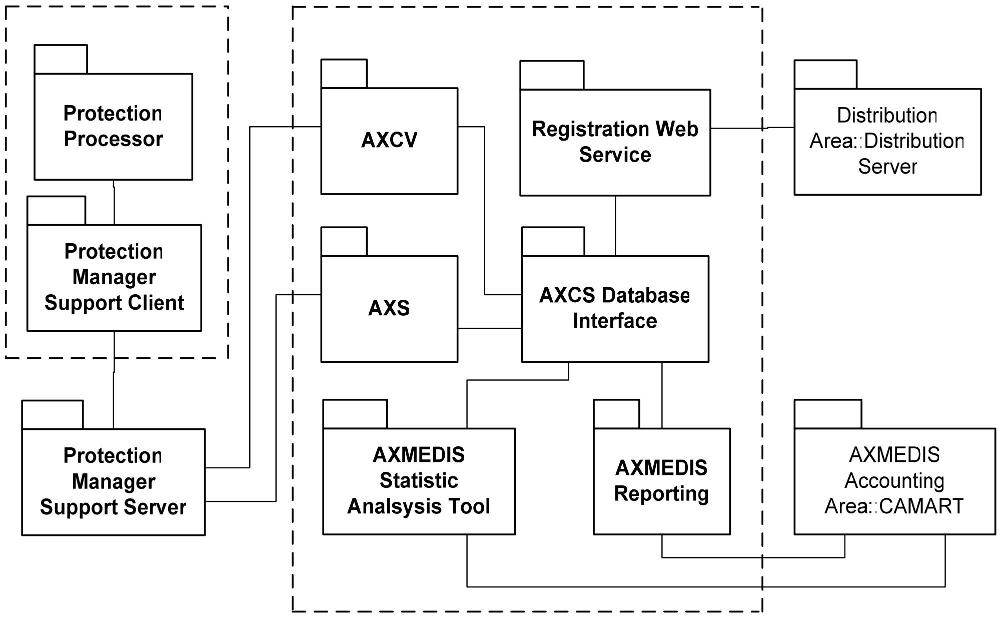

Protection Processor
Protection Manager Support Client
Protection Manager Support Server
AXCV
AXS
Registration Web Service
AXCS Database Interface
Distribution Area::Distribution Server
AXMEDIS Statistic Analsysis Tool
AXMEDIS Reporting
AXMEDIS Accounting Area::CAMART

Figure 61. AXMEDIS architecture regarding protection, rights management and accounting functionalities.

- **AXMEDIS Certifier and Supervisor (AXCS)**. AXCS is the authority in charge of user and tool registration (Registration Web service), user and tool certification (AXMEDIS Certification and Verification, AXCV), user and tool management (e.g. status supervision, automatic blocking, deadline supervision, etc.), user and tool unique identifier generation and object metadata collection. Regarding certification, AXCV uses an external Certification Authority (AXMEDIS CA) service which issues the corresponding user or tool certificates. This external CA is for the moment owned by the AXMEDIS consortium, but could be replaced by any other commercial CA service, if necessary. AXCS is also responsible for saving the Protection Information related to protected multimedia objects as well as the actions performed on them (AXMEDIS Supervisor, AXS), the so called Action Logs. Action Logs are the particular implementation of MPEG-21 Event Reports in the AXMEDIS context. AXCS also includes a user Registration service, useful for registering new users in the system from distribution servers. All these data are stored in the AXCS database, which is accessed though the AXCS database interface module in order to keep the access independent from its implementation. Other functionalities provided by AXCS are those related to reporting and statistical analysis, which are performed by the Core Accounting Manager and Reporting Tool (CAMART module) by analysing the information stored in the AXCS database and collected in Action Logs. The integral modules of AXCS (see Figure 61) have been developed as web services or libraries.

5.3.6.2 Securing User Tools

In order to ensure that users and tools are trustable inside AXMEDIS, several steps have to be taken in order to perform the registration, certification and verification of users and tools. In the next sections these steps are briefly described.

Registering and Certifying Users and Tools

All users of the system must be registered. If the user is a business user, offline verification has to be done in order to guarantee that the user is who he says to be. The user receives a certificate from AXCS that will be later used for the certification of tools being installed by him on a specific device. The user has also a user status in AXCS, which indicates if he has attempted some operation considered critical for the system that prevents the user from accessing the system or using the tools.

In order to register new tools, first, they must be verified to accomplish a series of guidelines. If the guidelines are not accomplished, then the tool is not registered and not usable inside AXMEDIS. On the other hand, if the verification is successful, then the tool is registered and AXMEDIS users can download and make use of it. During the registration phase, a fingerprint of the tool (software fingerprint) is estimated and stored in the AXCS database for later use. With this fingerprint, it is possible to check that the installed tool has neither been modified by the user nor corrupted after its installation.

Once user and tool are registered and verified, then the user is able to download and install an AXMEDIS tool on a device of their own. On the first use of the tool, the user needs to certify it. This process is done against AXCS, which is contacted by the tool. The user certificate is used in this phase for providing secure communication. AXCS checks that the tool fingerprint stored during the registration phase has not been modified and calculates the installed tool fingerprint, which contains information from the tool (software part) and the device (hardware part) where it is installed. This fingerprint is stored in AXCS so that during verification the tool fingerprint can be compared to the one registered during certification. The problem of using this mechanism lays on the fact that some characteristics of the user device may change (for instance, because some piece of the hardware may have crashed), which will invalidate the verification of the tool, making it unusable for the user. Later in this chapter, a possible solution to this problem, through reverification and recertification, is proposed.

Certification of Tools

The certification of a tool that is used in the AXMEDIS framework is a necessary step for that tool to work. Before a user is able to run and use a tool, the tool must connect to the AXCS to be certified as an “installed tool”. Before installation, AXCS verifies the tool integrity by comparing its fingerprint to the one stored during the tool registration process and, once installed, extracts some information (tool fingerprint) concerning the installation of the tool and the device where it is installed.

A malicious user who tries to certify a tool whose fingerprint does not match the original registered tool fingerprint would be automatically blocked in the system so that he cannot continue performing other operations within the system. Moreover the tool would not be certified and thus it would not be operative.

Once a user successfully certifies a tool, any user of the system who owns a valid AXMEDIS user certificate can use it. Blocked users cannot use tools in the system.

To perform the certification of a tool, the tool connects to the AXCS via PMS Client and PMS Server web service. In order to have a secure communication, the client certificate is used to authenticate the user against the PMS Server.

The certification process involves different operations in the AXCS:

- Generation of tool certificate and private key. AXCS Certification Authority generates a tool certificate. It is used to establish secure communications, via SSL providing secure web services, to the PMS Server by any user that manages the certified tool. In this way we ensure that only certified tools can interact with the server part in an authenticated manner.
- Generation of tool unique identifier. A tool unique identifier is assigned to that specific installation of the tool and is used to identify it when interacting with the server side. The identifier is generated following the UUID format [66] and inserted in the tool certificate.
- Generation of tool activation code. A tool activation code is used to enable the tool operation. Some cryptographic algorithms that depend on the specific installation are used to generate it and they are inserted in the tool certificate as a certificate extension.
- Generation of tool fingerprint. The tool fingerprint, as we have already said, concerns the installation and the device where the tool is installed. This fingerprint is used in further verification process to determine if the tool has been manipulated or if the device has changed or, in other words, to ensure the tool is still trusted in further executions.
- Storage of identifier, activation code, tool fingerprint and certificate. All the previous information is stored in the AXCS database and will be used to authenticate the tools that connect to the server part and to verify their integrity, as we will explain in next sections.

On the other hand, the certification process supposes also different operations in the client side (PMS Client and Protection Processor):

- Reception of tool certificate, private key, tool identifier and activation code. Regarding the tool certificate, private key, tool identifier and tool activation code, tool identifier and tool activation code are included in the tool certificate in the following manner (see Figure 62): 1) The tool unique identifier is used as the certificate common name (CN) in the subject distinguished name (DN) field; 2) The tool activation code is inserted as a certificate extension. The tool activation code extension is identified with the Object Identifier 1.3.6.1.4.1.25576.1.1, where 1.3.6.1.4.1.25576 is the Private Enterprise Number assigned by IANA to AXMEDIS Organisation and 1.3.6.1.4.1 corresponds to IANA-registered Private Enterprises [67]. Current assignment of the AXMEDIS tree corresponding to the 1.3.6.1.4.1.25576 branch is depicted in Figure 63. The tool certificate and private key are finally packaged by AXCS in a PKCS12 [68] structure protected with a password linked to the user that performed the certification and delivered over the secure channel established using the user and server certificates

```
Data:
  Version: 3 (0x2)
  Serial Number: 1000000493 (0x3b9acbed)
  Signature Algorithm: sha1WithRSAEncryption
  Issuer: O=AXMEDIS, OU=AXMEDIS AXCS CA, C=ES, CN=AXMEDIS AXCS
```

```
CA/emailAddress=axmedis@axmedis.org
  Validity
      Not Before: …
      Not After: …
  Subject:O=AXMEDIS, CN=ITO_cdecb4a1-dbcb-362c-a30d-bb936342996c
  Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
          Modulus (1024 bit): …
          Exponent: 65537 (0x10001)
  X509v3 extensions:
      X509v3 Subject Key Identifier: …
      X509v3 Authority Key Identifier: …
      1.3.6.1.4.1.25576.1.1.1: …
Signature Algorithm: sha1WithRSAEncryption
  …
```

Figure 62. AXMEDIS tool certificate fields.

```
1.3.6.1.4.1. 25576.0: reserved
1.3.6.1.4.1. 25576.1: AXMEDIS PKI-X.509 related objects
1.3.6.1.4.1. 25576.1.1: AXMEDIS Tool certificate extensions
1.3.6.1.4.1. 25576.1.1.1: AXMEDIS Tool activation code (or enabling code)
```

Figure 63. Assignment tree corresponding to the AXMEDIS IANA Enterprise number.

- Storage of certificate and private key and tool activation. The PKCS12 structure is accessed by Protection Processor in order to extract the tool certificate and private key, which are finally stored in a local keystore, and also to get the activation code used to enable the tool.

Verification of Tools

Verification of tools is devised to cover two functionalities. First, it provides a means to ensure that client tools have neither been manipulated nor corrupted. Moreover, verification is used to resynchronise all the actions performed by users during offline operation, which were stored in the local secure cache.

Verification of tools is performed periodically by the Protection Processor and every time the user tool resynchronises the offline performed actions with the server part. It consists on the verification of the estimated tool fingerprint in the moment of the verification against the tool fingerprint stored in AXCS database during the certification of the installed tool.

Regarding the tool integrity verification, if AXCS detects that critical parts of the tool or the device have been manipulated, it can adopt the pertinent measures as, for example, blocking the specific installed tool for which the verification failed.

Regarding the resynchronisation of offline performed operations, AXCS executes an algorithm to determine whether the received list of operations, which are called Action Logs in the AXMEDIS context, is complete with respect to the previous received operations. This integrity check is feasible thanks to the calculation of a fingerprint on the performed Action Logs, which is computed by PMS Client during the tool operation. This fingerprint is sent to AXCV when resynchronising the offline Action Logs and is verified by AXCV using the algorithm depicted in Figure 64.

The history fingerprint (FP) computation is also performed in the client side for each action performed in the online or offline operation, so that, once synchronised it must hold the same value in both PMS Client and AXCS. When an online operation is performed, this value is immediately synchronised in the server side. When any offline actions are performed, an Action Log associated to each of them is stored in the local secure cache, where PMS Client computes and separately stores FP value after the operation.

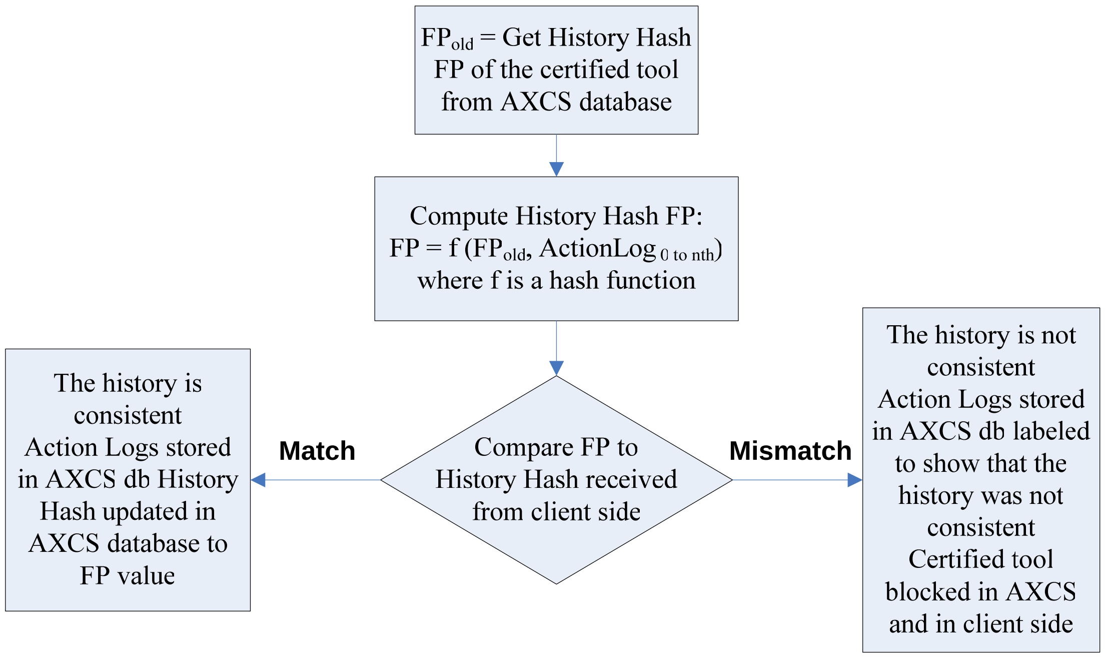

FP_{old} = Get History Hash FP of the certified tool from AXCS database
Compute History Hash FP: $FP = f(FP_{old}, ActionLog_{0\ to\ nth})$ where f is a hash function
Compare FP to History Hash received from client side
Match
The history is consistent Action Logs stored in AXCS db History Hash updated in AXCS database to FP value
Mismatch
The history is not consistent Action Logs stored in AXCS db labeled to show that the history was not consistent Certified tool blocked in AXCS and in client side

Figure 64. Algorithm to determine the integrity of the received Action Log list in AXCV.

Recertification of tools

Recertification functionality will be accessed by tools for which some parts of the hardware have changed after the installation of the tool. These changes will be only detected and therefore taken into account if the hardware parts changed are included in the tool fingerprint calculated during the certification phase. Thus, a change involving tool fingerprint makes the verification (and reverification) phases fail. Reverification consists on checking the complete fingerprint of the tool instead of a hash of the fingerprint (used for speeding verification phase). Reverification will start the recertification process of the tool, if the detected changes are considered acceptable.

After AXCS detects the inconsistency in the hardware tool fingerprint between client and server sides, it has to decide if the hardware changes, as well as the operating system changes, done by the user are acceptable. If so, AXCS will request a new certificate to the AXMEDIS CA for that installed tool reusing the tool identifier, but recalculating the hardware fingerprint and enabling code associated to this tool. However, if AXCS determines that hardware changes are not acceptable, reverification will fail and the tool will be blocked.

In order for AXCS to be able to determine if a new certificate can be issued after hardware changes, several information has to be stored. This information includes acceptable changes, acceptable timing for changes and so on. In the rest of this section, we outline the information that needs to be stored and also the problems associated to its storage and value definition.

Hardware changes

When considering the hardware changes that can be accepted in a user device a question arises: which changes are permitted for allowing the recertification of a tool? First of all, we should analyse which information is part of the hardware fingerprint of the device. Then, decide if it is acceptable to change it to allow the generation of a new certificate. If so, the last thing to decide is the frequency of the changes allowed to the user.

Table 13. Allowed Hardware Changes and Frequency.

Hw Change	Frequency	Considerations
Disk where tool is installed	Not acceptable	All information regarding the installation of the tool will be lost (secure cache, history, etc.). Possibly also the tool will be lost.
Disk where tool is not installed	Once a year	--
Network card	Once a year	If there is more than one network card, the times for changing them should be considered separately
CPU	Once a year	If there is more than one CPU, the times for changing them should be considered separately
Operating System (OS) Name	Not acceptable	--
OS Serial	Not acceptable	--
OS Upgrade	No restriction	--
OS Version	No restriction	--
BIOS	Once a year	--
Network card + CPU + BIOS (Integrated)	Once a year	--

It is obvious, as shown in Table 13, that frequency restrictions for hardware component changes are quite arbitrary, as, if some hardware component is defective and the user has to change it several times in a year, their tools may not be recertified. Moreover, this solution has the drawback that some changes may be not detected by the server side (for instance, if the disk that has been changed is the one where the tool was installed or not), having as a consequence the tool not being recertified.

Nevertheless, the proposed recertification solution will allow non malicious users with hardware problems to continue using their AXMEDIS tools in some situations in a transparent manner. Another option would be not to recertify any tool, which is the current situation, forcing the user to reinstall the tools again, with the consequent loss of time for him (and maybe also the access to the content he had already purchased).

Finally, the allowed frequency restrictions information needs to be stored in the AXCS database for permitting its checking after a reverification failure.

5.3.6.3 Use Case

In this section we present a scenario to illustrate how the proposed architecture and the processing of protected and governed multimedia content are related. It describes content consumption of this protected and governed multimedia content.

Imagine that a user has purchased online a license that grants them the right to play a movie during a certain period of time. The acquisition of the license could be performed in various ways. On the one hand, the user could have obtained the license in the same place where he purchased the content. In this case, if the license needs to be customised for a particular user, the content distributor must request the license to the corresponding protection and governance servers (for instance, PMS Server and AXCS). On the other hand, the user could have obtained the content through a P2P network, or other online or even offline distribution channels. In this latter case, the content must have some metadata that identifies the content server (associated to a governance server) with which the user must interact to purchase a license.

The aforementioned user has, installed in their device, a specific tool or plug-in, for example, AXMEDIS player, which manages the protected and governed objects of the proposed system and which is able to display them in the appropriate way.

The use case begins when the user downloads a protected and governed movie, opens it with his favourite player, which includes the appropriate plug-in and tries to watch it (Play movie). Although the plug-in has not been manipulated, the system needs to verify its integrity and certify it before allowing its operation (which includes the unprotection of the content, for playing it).

Figure 65 shows the steps involved in the content consumption use case, which are the following:

1. The viewer requires unprotecting the movie to an internal trusted module, Protection Processor.
2-3. Protection Processor estimates the installed tool fingerprint and connects to AXCS through PMS Client and Server in order to certify the tool.
4-5. AXCS successfully verifies user data and status and tool integrity with respect to registered tool.
6-7. AXCS stores in its database the estimated tool Fingerprint, which includes the software part of the user tool as well as a hardware part and computes a unique tool identifier.
8-10. AXCS requests the AXMEDIS CA the tool certificate by passing it the tool enabling code. The CA generates the certificate with the enabling code as an extension.
11. AXCS sends Protection Processor a PKCS12 structure that contains tool private key and tool certificate with the tool identifier and activation code.
12-15. Protection processor stores tool certificate and private key in a local repository, extracts activation code and enables tool operation.

Now suppose that the user, after making his first content consumption attempt, needs to change some hardware element of his device, as for example a hard disk where the tool is not installed or his network card. After this, when trying to do an action with the tool, the steps would be the following:

16. Before the authorisation, Protection Processor always calls verify method to check tool integrity and resynchronise offline Action Logs. In order to call it, it must reestimate the tool fingerprint and extract user and tool information from pertinent certificates.
17-19. PMS Client gets action logs from secure cache and contacts AXCS through PMS Server. (Note that in this case, as it is the first usage, there will not be any action logs)
20-24. AXCS verifies user and tool data with respect to the certified tool Fingerprint, computes and verifies the operation History Fingerprint and stores received action logs in the AXCS database. The tool fingerprint hash does no match the one in AXCS database.
25. The negative result of the verification is sent to Protection Processor.
26. Protection Processor asks for reverification by sending the whole tool fingerprint in XML format instead of only the tool fingerprint hash.

27-30. AXCS verifies user and tool data with respect to received tool Fingerprint. It detects that the software part of the tool fingerprint has not changed, but that there are some changes in the hardware part of the tool fingerprint.

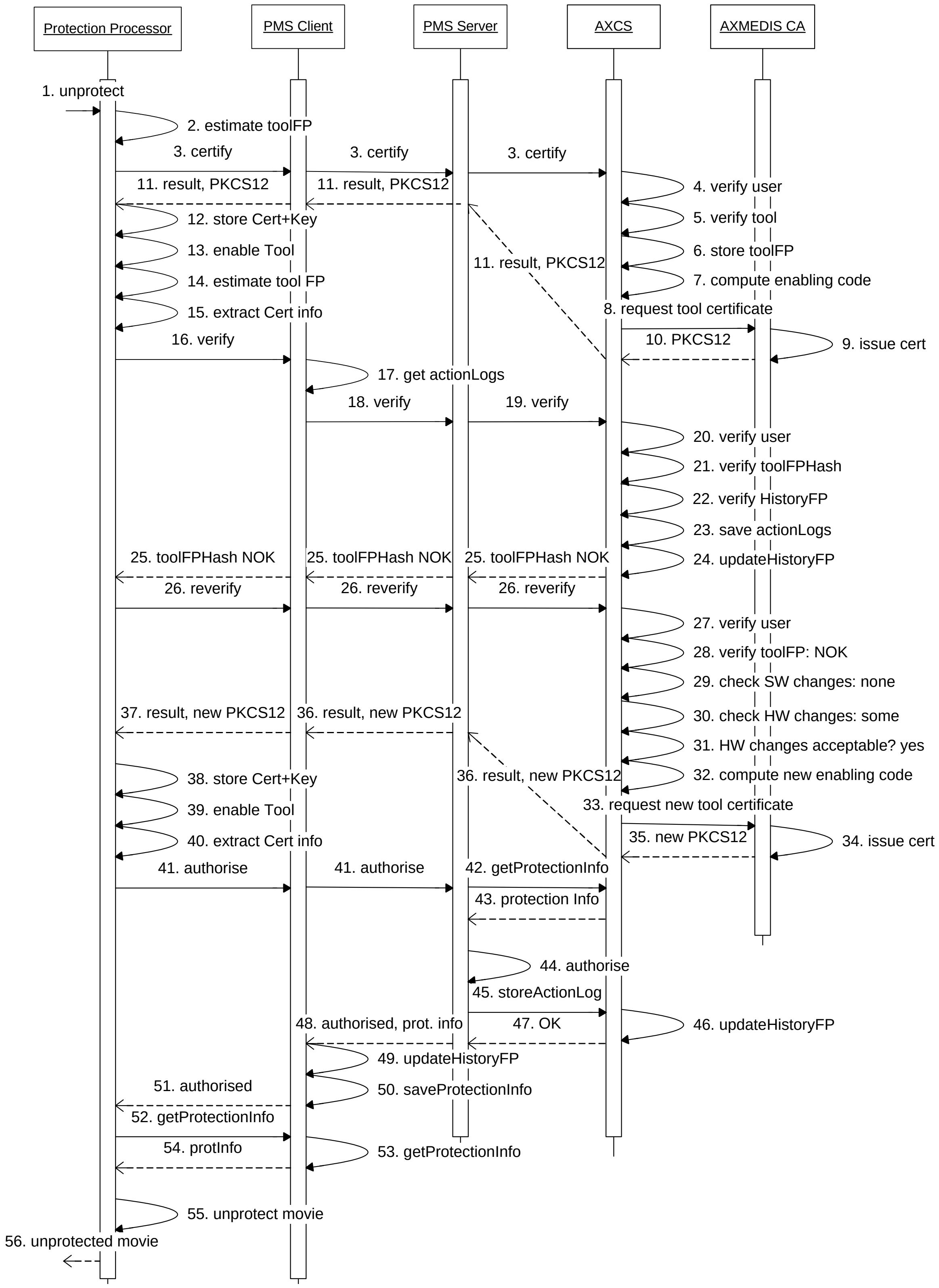


Figure 65. AXMEDIS Use Case.

31-35. For the detected hardware changes, AXCS determines whether they are acceptable or not in the system. If so, it computes a new enabling code for the tool and asks AXMEDIS CA for a new certificate and private key for that tool.

36-37. AXCS sends Protection Processor the new PKCS12 structure that contains a new tool private key and tool certificate, the old tool identifier and a new activation code.

38-40. Protection processor stores tool certificate and private key in a local repository, extracts activation code and enables tool operation.

41. Protection Processor asks for authorisation and for protection information to PMS Client. As the user is working online, PMS Client contacts PMS Server.

42-43. PMS Server contacts AXCS to retrieve the object protection information.

44. PMS Server performs the license-based authorisation using its license repository.

45-47. As the authorisation is positive, PMS Server sends the pertinent Action Log to AXCS, which stores it in its database.

48. PMS Server notifies PMS Client the successful authorisation and delivers the protection information to PMS Client.

49-50. PMS Client updates and stores the operation history hash fingerprint and the object protection information in the local secure cache.

51. PMS Client notifies Protection Processor the successful authorisation.

52-54. Protection Processor requests the Protection Information to PMS Client, which retrieves it from the local secure cache.

55-56. Protection processor is capable of unprotecting the protected object so that the player can finally display the film to the user.

It is worth noting that, once the tool is certified or recertified, the verification process is done only when the user wants to consume multimedia content or at arbitrary times decided by Protection Processor. Steps 2 to 14 are no more executed after the tool certification.

It is also important to observe that the protection information used to unprotect an object is only delivered to the client application after a successful authorisation, as an atomic operation. In this way the system only enables authorised users have access to protected objects.

5.3.6.4 Testing procedure

In AXMEDIS, we have defined a procedure for testing the software developments corresponding to the different project partners. As in the AXMEDIS framework there are several dependencies between software components, several steps have been taken into account for the software testing. The software testing procedure is based on the definition of test cases that correspond to the use cases defined to satisfy the project requirements. Starting form the defined test cases, the testing methodology involves unit testing, regression testing, integration testing and acceptance testing. Other testing steps such as those related to the performance of the components have not been included in this section, as they do not correspond to the work presented in this thesis.

Test Cases

The test cases will be specified for each of the components of the AXMEDIS framework by using the form presented in

Table 14. The model will be UML including: name, ID, description, functionality to be tested, context, partners involved, validator(s) skill, data set needed, steps, expected results, variations, issues, additional activities to be considered, metrics to be used, etc.

Table 14. Structure of a Test Case.

TCId	Unique identifier of the test case
Test case	Name of the test case
Initial conditions	Description of the state of the system before the execution of the test case. This state is the one needed for the correct execution of the test case
Configuration description	Description of configuration conditions, tools involved and connected
Description of functionality to be tested	Functionality to be tested
Partners, people involved	List of people involved in the test, partners, user-groups, other people needed
Validator(s) skill	Skill of the people involved in the test during the validation with end-users
Data set used	Names of or references to the data sets used or their number
Steps	Steps of the test
Expected results	Expected results of the test
Variations	Some changes that can be done for testing some slightly different functionalities
Issues	Other issues, notes, annotations if the Test Case is not clear
Test case Scope/Type	The applicability scope of the test case, such as GUI, backend, etc and the type of the test BlackBox, WhiteBox, UnitTest, and so on

Component Validation

The component validation process involves the checking and analysis of the component in order to verify that the component meets the requirements.

The main validation activity is the testing of the component. The main steps in the testing phase are the following:

- During the development phase of the component, unit tests should be done in order to check that the functionality being implemented is the expected. This task should be done by the development team of the component, as they have the complete knowledge over it. Any modifications with respect to the specification should be reported.
- When the development of a component has been completed, an initial unit testing phase and an integration test is needed to evaluate how the unit performs and how it interacts with other units. This task should be done with the cooperation of the development teams of the integrated components. The result of this task should be reported in order to perform the needed actions for solving the problems found. In this task the following aspects need to be evaluated and reported:
    - Documentation provided
    - Differences with respect to the specification, if any

    - Interfaces amongst components

- The rest of tests should involve the whole system and will differ depending on the demonstrator being evaluated.

In order to perform the above tests, a set of test cases has been identified and described, together with the necessary content.

Apart from tests and reviews, component validation also includes the examination of the degree to which the component is properly documented, including updates in the specification documents, documentation of the programming interface, usage manual (specially for end-user or business-user applications) or installation manual.

Component Acceptance

Component acceptance mostly involves the components that are addressed to user applications, either for business or final users.

Acceptance testing allows users, to test the functionality of the system against the requirements and use cases. Each kind of tool needs to be tested by a key user In the area. For instance, the component acceptance test of a content production tool should be done by a user that is skilled in the area together with part of the development team, in order to get the comments on the tool behaviour.

It will be very useful to follow the test cases that involve components to be accepted in order to check if they are correct. A report on if test cases are accomplished or not should be done by the test participants.

The acceptance of a component means that the component has been previously validated and verified, making the corresponding unit and integration tests. Once those tests have been passed, the acceptance tests should be done. Acceptance testing involves other partners different from those who developed the component, such as final users, user group experts, etc.

Acceptance test results could be reported using the review form described in Table 15.

Table 15. Review form for acceptance and integration tests.

Review ID	Identifier of the review.
Module names	Name of the modules being reviewed.
Module description	General purpose of the module
List of use cases	Related Use Cases
List of Test Cases	Related Test Cases
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.
Participants	Names, organisations and roles of people involved in the review.

ID	**Who**	**Date**	**Issue location and description**
Issue number	Name and organisation of the person who finds the issue	Date when the issue is found	Part of the module where the issue is found and description of the issues found in the module during revision and. The location could be the source code file, web page, etc; it will depend on the type of application.

After a report has been defined, the verification by technical people need to be performed. Next we give an example of what has to be reported when a module is verified, after it has been reviewed and some issues have been found. The information to be reported is described in the following table.

Table 16. Verification form for acceptance and integration tests.

Verification ID	Identifier of the verification.
Review ID	Identifier of the review to which this verification refers.
Module names	Name of the modules being verified.
Author	Organisation that has the responsibility of the implementation of the module. The name of the person involved in the implementation can also be given.

ID	**Who**	**Date**	**Status**	**Issue description and resolution**
Related Review Issue number	Name and organisation of the person who verifies the issue	Date when the verification is performed	Solved, Proposed or Not Solved	How the issue found during the revision process has to be solved or has been solved

Regarding the Status column, it has to be filled in the following manner:

- Solved: when the issue in the review report has been solved
- Proposed: when the issue in the review report has been proposed but not solved
- Not solved: when the issue in the review report has been neither proposed nor solved

Once the component has been accepted, it can be made publicly available in the AXMEDIS framework. Moreover, it may be further optimised.

Component Validation and Acceptance

In order to start up the component validation and acceptance in the AXMEDIS framework, the responsible of each module needs to:

- Follow the CVS repository guidelines.
- Prepare unit tests.
- Perform unit tests and give a brief report on them.
- Prepare integration tests.
- Perform integration tests. Integration tests are sometimes done by the partners participating in the integration together but sometimes not. In the latter case, some communication mechanisms will be established in order to solve possible integration problems as soon as possible, as e.g. videoconference, audioconference, e-mail interchange or chat.

Periodic Verification and Regression Testing

During the project development, components will be improved and updated to solve the problems found during the different testing phases or to meet new requirements found during the acceptance tests or the evolution of commercial software products and standards affecting the AXMEDIS framework components.

In this way, there will be the need to inform the rest of the partners about the updates in any components, so that new integration tests are performed if needed.

Reports on the unit tests performed over the new or updated components have to be given together with the new version of the component.

The periodic verification steps should be as follows:

1. Update component into the corresponding directory of the CVS repository, indicating that it is a new version.
2. Send e-mail to the reflector and / or developers mailing list to inform that the update of a component has been done and including the results of the related regression test(s) when needed.
3. Partners using the updated component should:
    a. Perform integration tests with the new version of the component. The participation / support of the component responsible may be requested.
    b. Report errors / problems detected during the integration test, if any. The report has to be uploaded in the portal, in order that partners are informed of the results of the test.
    c. If needed, after integration errors / problems reported in 3.2 are solved, components using the initially updated component should be also updated in the CVS repository.
    d. Go to verification step 2 for the component(s) updated in step 3.3.

As a modification in one component may involve the modification of many other components, regression and integration tests should be systematically done. In some cases, it could happen that a component will no longer work with lower versions of libraries and components. This should be specified in the documentation of the component.

Integration testing in AXMEDIS

This section aims to describe how to validate the AXMEDIS Framework by providing the tools or procedures to perform an integration testing of the different modules.

For this purpose, it is assumed that a Unit and Regression testing initial step has been already performed on the AXMEDIS modules, so that what has to be tested here is not the correctness of the modules operation but the interaction and compatibility with other modules.

Next, we describe the general procedure to be followed for all the AXMEDIS modules, that is, a general description of the modules with which the module integrates, a description of how integration testing needs to be done and a standard means for reporting the errors found during a review plus the reports used to describe the proposed or implemented solutions.

For each of the AXMEDIS modules the following information should be provided:

- Modules with which it integrates. A list of modules that interact directly or indirectly with the present module and against which some aspects have to be verified to have a successful integration of the AXMEDIS Framework.

- How to perform Integration Testing. A reference to the tools that have been created for the automatic or semi-automatic testing of the integration with other modules or, when automatic tests become difficult to be developed, the description of the procedure to be followed and the aspects to be verified during the integration testing.
- Review report form. The reports regarding the results obtained during the review of the integration of different modules. Each report describes the issues and errors found during the review. The main information to be reported when a module is being reviewed is described in Table 15 and Table 16.

Another useful mechanism for testing the integration has been the generation of the AXMEDIS Tools (AXTools). The AXTools consist of an installshield which includes all major AXMEDIS Tools that can be freely distributed for being installed and used by AXMEDIS partners as well as non-partners. The production of the AXMEDIS Tools has become an important means for proving the integration of the modules in the AXMEDIS Framework as well as the interaction of different AXMEDIS applications.

6 Linked-Work: an architecture for the management and protection of intellectual property

6.1 Motivation

Once we have defined and seen how a DRM architecture such as MIPAMS can be applied to different contexts, we will now present a specific implementation, which extends in some way the functionality previously proposed. The goal of this new architecture called Linked-Work is to facilitate the monitoring of a work usage by its author or authors. The main idea behind this objective is to keep a link between any work, which is represented in a digital format by a digital object (also called object representation), and all the works that derive from it. The link will be in some way embedded in the object representation, enabling a direct way to discover the ancestor, and recursively, the whole lineage of a given work. We will see how this link also enables advanced functionality.

Linked-Work is an innovative architecture that enables the registration of original and derived digital content while providing some Digital Rights Management features that enable the creation of trust chains along the multimedia content value chain. Linked-Work system combines different features common in DRM systems such as licensing, content protection, authorisation and reporting together with some innovative concepts, as the linkage of original and derived content, the definition of potential rights and the revocation of the offered rights. The transmission of reporting requests along the content value chain combined with the possibility for the authors to preserve some rights over the derivative works enable the system to determine automatically the percentage of the incomes corresponding to all the actors involved in different steps of the creation and distribution chain. Linked-Work consists of a web application which interacts with different external services plus a user application that can be used to manage protected contents.

The research work and results presented in this section have submitted to the following International Conferences:

- ACM International Conference on Multimedia 2008 [83].
- Fourth Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS’08) [73].

Both publications are very recent and are still acceptance pending.

6.2 Architecture and Integral Components

Linked-Work is a service-oriented architecture that relies on a distributed functionality paradigm. It consists of a main web application, accessible through a web browser, which interacts with different DRM components that are implemented as web-services. It also includes a user desktop application which deals with the rendering of protected content.

Figure 66 depicts the Linked-Work overall architecture.

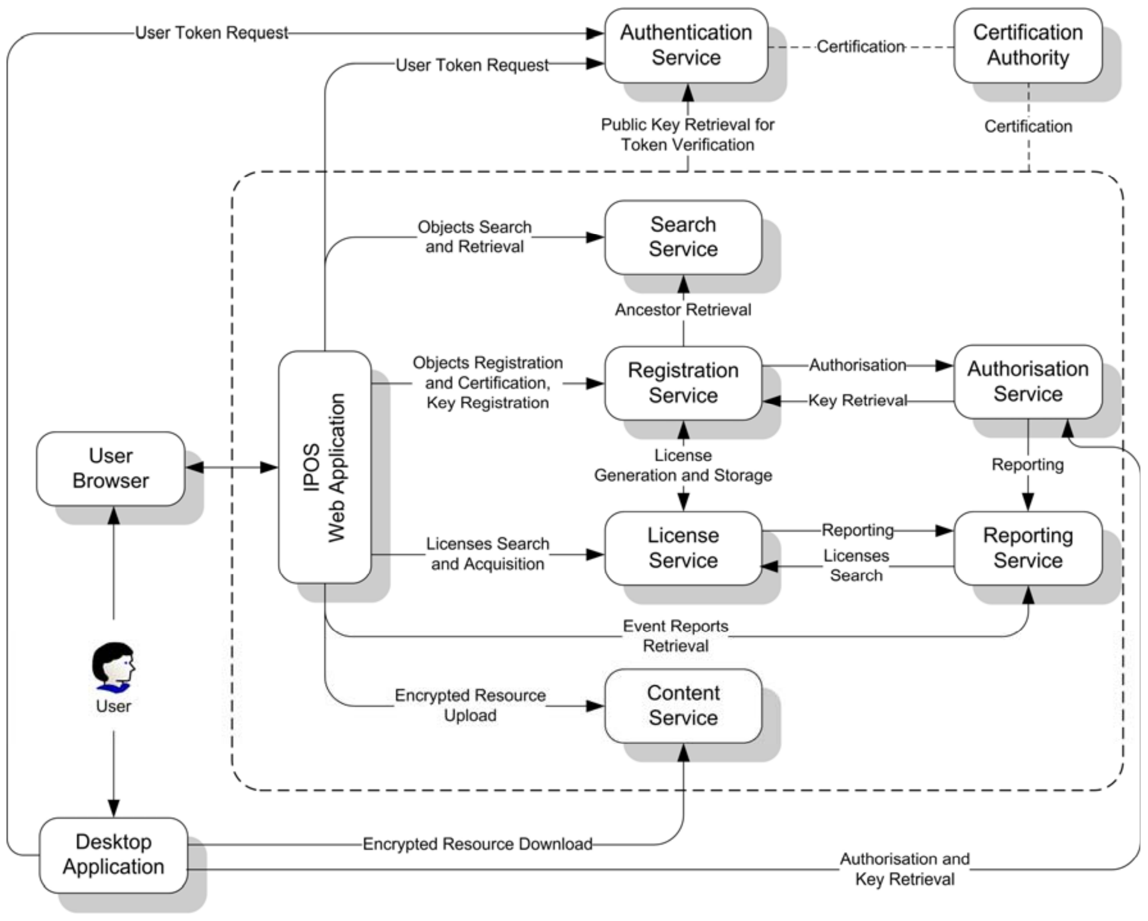


Figure 66. Linked-Work architecture.

All the services depicted in Figure 66 are provided by the Linked-Work system itself. However, some of them, such as the Search service, License service, Reporting service, Content service and others may be external, as far as they have access to the information stored in the system's databases.

The information in the system is stored into different databases, keeping the access banned for the web application and user application and being only authorised for web services. Linked-Work involves the following databases:

- Objects database: xml of the object, signature and resource protection keys
- Objects Information database: metadata fields for efficient object searches
- Reporting: xml of the event reports and metadata fields for efficient searches
- Payments: issuer, recipient, fee amount, fee currency
- Users: personal data and account status (acceptance pending, accepted, cancelled)
- Licenses and context: license anonymous templates for acquisition, personal licenses and status (correct, revoked) and number of times the right in a license has been exercised.

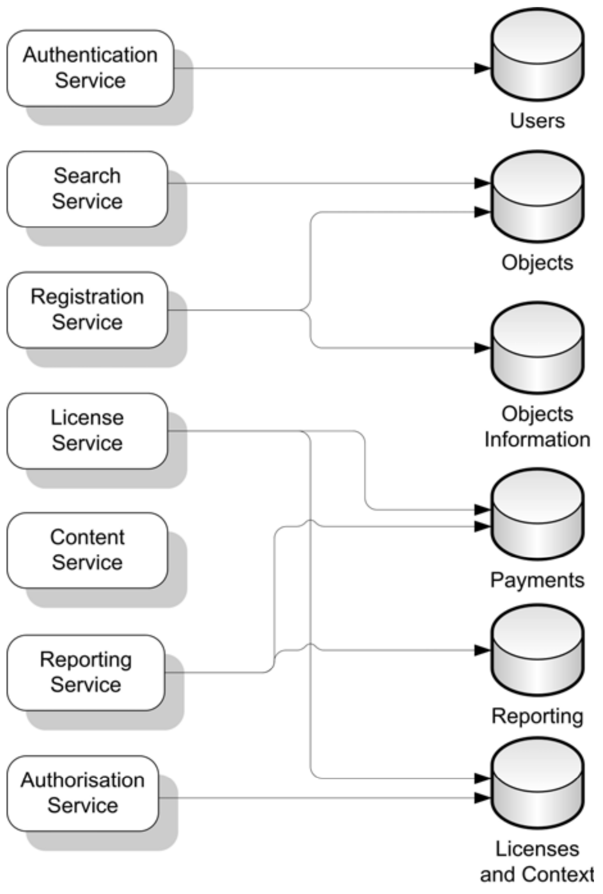


Figure 67. Linked-Work databases.

In subsequent sections we will describe in detail which are the functionalities provided by all the components in the system.

6.2.1 Services

User Authentication and registration service

The user authentication and registration service acts as a single-sign-on access point, by issuing SAML tokens for any user or service in the system. By means of these digitally signed tokens, users can authenticate in the system and services can authenticate themselves against other services. More in detail, the service's functionalities involve the following:

- Registers and manages users and user Groups. It offers the necessary methods for inserting new users in the system and modifying their status. User accounts can be in different states:
  - Acceptance pending. This is the default state when a user is registered by means of the registration form.
  - Accepted. This state is reached when the system administrator validates and accepts the user request.
  - Cancelled. This state is reached only if a system super-administrator cancels the user account due to an inappropriate usage or violation of the terms of use.

- Authenticates users and user Groups in the system. Given a user name and password, this service returns a SAML token which can be used to authenticate the user against any service available for users. Any service called from the web application or user application will receive and verify the user token before doing any other operation. Additionally, the user token will be used to enable the interaction between services, that is, any service in the system that requires calling another service will send the user token, which will be the key to access the other services' functionalities. In this way, any operation in the system is executed only if the user is authenticated and there is no way to perform operations on behalf of another user.

Certification service

The Certification service is responsible for certifying by digitally signing the objects received from the registration functionality in the main web application. In detail, it includes the following procedures:

- Receives object representations in XML format, which include the object's metadata, potential rights, resource hash, resource reference and Event Report Requests following the MPEG-21 standard.
- Calls the authorisation service to determine if the user owns a license that authorises them to exercise the requested operation.
- Generates a unique identifier for the object and inserts it into the object. The object's unique identifier follows the UUID format, and is of the form URN:LW:DCI:D5A61528-9266-3621-9045-9B7E2F94AE99.
- Inserts the object's parent unique identifier into the object. This feature is very useful for being able to track the source of the object by providing a link to the parent object.
- Digitally signs the object's XML document by using XML signature. All objects in the system are digitally signed by means of XML signature and embed the signing certificate. In this way, all objects can be checked for authenticity and integrity anywhere, as long as the certification service's X.509 certificate, which contains the service's public key, is made public.
- Saves the certified objects into the objects' database.
- Saves the object's metadata fields into a specific database by means of the search service. The object's metadata fields are parsed and stored, thus enabling an efficient way for searching according the metadata fields.
- Requests the generation of anonymous potential licenses or license templates. The potential rights defined by the object creator, which are included inside the objects, are transformed into anonymous licenses which are made available for acquisition through the web application.
- Generates and sends the corresponding Event Reports to the reporting service, following the MPEG-21 standard, for all the authors involved in the content value chain.

Content Registration service

The Content Registration service consists in an application that depends directly on the Linked-Work web application. It uses sockets instead of standard Web service calls for a better transmission performance. Some of the operations performed by this service are the following:

- Receive the resource name and encrypted content bytes by means of a socket connection.
- Store the content in a local path.

Search service

The Search service provides a means for retrieving objects when searching amongst the objects' metadata. Mainly, this service:

- Returns a list of objects stored in the system database which match the provided searching criteria.
- Receives objects' metadata to be stored into the search service database.

The searching functionality is divided into three specific types of searches:

- Global search: the provided results are obtained from searching amongst all the objects in the system.
- Own search: the provided results are obtained from searching amongst all "own" objects in the system, i.e. those registered by the authenticated user.
- Acquired search: the provided results are obtained from searching amongst all "acquired" objects in the system, i.e. those registered by any user different from the authenticated user for which the authenticated user has acquired at least one license.

License service

The License service deals with the generation and archival of licenses, which are used when asking for authorisation. In detail, this service:

- Returns the collection of licenses which can be acquired by any user for a specific resource. This functionality is useful for presenting the users the licenses may like to acquire.
- Customises licenses available for acquisition for a specific user or user Group. Once a user has selected a set of rights and conditions from those offered by the author of the object, this service generates the specific instance of those rights for that user and stores the resulting license into a database, which is later used for authorisation purposes. This license grants the user to exercise some rights over an object under certain conditions. It can be used to prove that the user has acquired some rights.
- Returns the collection of licenses which apply to a specific user and resource. This functionality is useful for presenting the user the licenses they own over a specific object.
- Returns the number of times a user license has been used. This functionality is useful for presenting the user the number of executions they have made and inform about the remaining in case there is a usage limit condition in the license.

Authorisation service

The Authorisation service enforces the fulfilment of the rights and conditions expressed in licenses. It answers the following question: is user U authorised to exercise the right R over the Object O? For that purpose, it searches the applicable licenses in the license service to determine whether the user is allowed or not. In short, it:

- Performs the authorisation of the action requested by the user over the requested object according to the conditions present in the licenses owned by the user.
- Generates the corresponding Event Report, following the MPEG-21 standard, towards all the actors of the content value chain and sends it to the collecting reporting service. Event Reports are generated when the user is authorised to perform the operation as well as when they are not

Reporting service

The Reporting service collects the reports about content usage, while providing searching capabilities amongst the collected reports. It acts as follows:

- Receives reports generated by the system (e.g. certification and authorisation services) and collects them into a specific database
- Provides searching functionality by returning a list of Event Reports that match the requested searching criteria, which may be any of the event report fields.
- If the received report entails a payment duty, the reporting service determines the type of payment. If the type is FEEFLAT, which means that the payment is performed at the moment the license is acquired, is does nothing, as the payment is controlled by the web application when the license acquisition is executed. However, if the type is FEEPERUSE, which means that the payment needs to be cleared every time the license is used, the service determines how the payment fee needs to be propagated along the value chain to fulfil not only the license terms but also the rights over derivative conditions present in the license of other ancestor objects in the value chain. The payment information is stored in a specific payments database, which can be useful in the case where the society who manages the framework decides to act as an intermediary, by collecting the payments from some users and redirecting them to their addressees.

6.2.2 Applications

Web application

The Linked-Work main application consists in a web application from where the user can access almost all the system functionality, available in different sections:

- Registration of new Objects. In this section the user can register any kind of Object, including new Works, Licensed Works, Work Manifestations, Work Manifestation Copies, Adaptations, Adaptation Manifestations, Adaptation Manifestation Copies, Instances and Copies. The web application provides a form where the user can fill the metadata field of the objects, define the potential rights and attach a resource, when needed. It also gives the author the possibility of receiving reports about the usage of their content by selecting a checkbox. If so, every time a user performs an action over one of their objects, the author will receive a report describing what has happened. It makes use of the Certification and Content Registration services. For design restrictions, the servlet part of web application is responsible for receiving the resource, generating an encryption key, encrypting the content, storing the protection information in a specific database and make use of the Content Registration service to store the resource in an optionally external resource repository. However, this protection functionality could be moved to the Content Registration service.

- Search amongst own Objects. In this section the user can search by any of the metadata fields of the objects that were registered by them. It makes use of the Search service.
- Global Object retrieval and download. Similar to the previous case, here the user can perform a global search amongst all the objects registered in the system by any user. When the resulting objects are presented to the user, several options are made available, as e.g. view the object's XML, download the object's XML, navigate towards the object's parent (from which current object derives) if available, and acquire a license. It also makes use of the Search service.
- License acquisition. This option is accessible from the results obtained in previous step. It is only available when there are any rights to be acquired which were defined by the original author as potential rights. When the user selects this option, they are redirected to a web page were they can select the rights and conditions amongst the different options the original author made available. Once the user selects the right and conditions they are interested in, the web application contacts the License service in order to generate the corresponding license for that use. Currently, license acquisition does not involve the payment of the fees that may be specified in the potential rights, but a payment commitment that eventually needs to be fulfilled.
- View acquired objects. Once a user has acquired some licenses that enable them to exercise a right over an object, as reported in previous step, they can consult all of them in a specific section of the web application. This section also enables the user to register derivative objects from those for which he owns a license that grants them the corresponding derivation right (e.g. makeAdaptation, makeInstance, etc.).
- Search and view the Reports generated towards the user. The web application includes a section where the user can consult all the reports collected by the Reporting service which are directed to them.
- Personal data management. The web application provides a section which enables the user to modify their personal data, except their name and surname. It interfaces the authentication service.

Player

The Linked-Work Player is devised for rendering the resources associated to some kinds of DCIs while ensuring the enforcement of the rights and conditions established in the license acquired by the user. Current development enforces the rights and conditions fulfilment in the moment when the resource is tried to be rendered. However, it opens the resource with the system's predefined application, so that it is shown in clear to the user.

Next, we summarise the main functionalities available from the player:

- Load Object. The player opens the object's XML document and displays the metadata to the user. This functionality is available for all kinds of users, not only those already registered in the system but also those not registered, for any object they have in their local machine.
- Link to the web application. The player offers the user the possibility of enrolling the system by providing a link to the section of the main web application devised for registering users. It also redirects registered users to the web application section were they can acquire a license for the object that is loaded.

- Download the encrypted resource associated to the DCI. The player enables authenticated and authorised users to download the resource associated to the loaded object. The resource is encrypted using an AES block cipher (symmetric key cipher), so it cannot be accessed by the user unless decrypted.
- Decrypt and render the resource if the user is authorised to. The player asks the user to select the operation they want to perform and asks the Authorisation server for authorisation. If successful, the player will get the encryption key that can be used to decrypt the resource and render it.

6.3 Relationship with MIPAMS

The Linked-Work system is based, in general, on the MIPAMS architecture. However, some components have been decoupled or simplified in order to make them more specific in terms of functionality. Table 17 provides the mapping between the components defined in the MIPAMS architecture and those in Linked-Work.

Table 17. Linked-Work mapping with MIPAMS.

MIPAMS component (functionality)	Linked-Work component
Governance server (license generation) Governance server (authorisation)	License service Authorisation service
Supervision server (reporting) Supervision server (authentication) Supervision server (certification) Supervision server (verification)	Reporting service Authentication service Not present Not present
Adaptation server	Not present
Content server (object registration) Content server (content search) Content server (resource upload)	Registration service Search service Content service
Protection server (resource encryption)	Linked-Work web application
Registration server (user registration)	Authentication service
Certification Authority	Certification Authority
Trusted client	Desktop application
Intermediary (client side) Intermediary (server side)	Desktop application Linked-Work web application

It is worth noting that the intermediary component defined in MIPAMS is included in Linked-Work both in the client and server parts. On one hand, the Linked-Work Desktop application includes an intermediary, as it interacts with different services at the same time, such as Authorisation and Content services. On the other hand, the Linked-Work Web application can be also seen as an intermediary that orchestrates the interaction of the user with all the services in the system.

Moreover, Linked-Work does not make use of the certification and verification functionalities defined in MIPAMS, as it has not been considered necessary for being implemented. Thus, Linked-Work user applications are only enabled to operate in an online mode, and the interaction with other services, which is made by means of SAML token authentication, is considered secure enough so as not to implement the MIPAMS certification and verification functionality.

6.4 Key Concepts

6.4.1 Value Network, User Roles and Rights

The creation and distribution of content (either digital or physical) necessarily involves the intervention of different user roles (from the creator to the end user), which determine the nature of the value network associated to the content being modified and/or exchanged. In this section we will focus on the creation value chain, describing the user roles that participate in it and the operations that they can perform over the content during its lifecycle. The definition of the creation value chain is based on the Digital Media Project (DMP) [18] Creation Model [69].

DMP defines the creation model describing the different user roles participating in the different steps involved in the creation of content, to which one or other types of Intellectual Property (IP) known as an IP Entities may be associated. The steps of the creation model define possible value networks associated to content creation, determining how the digital content evolves from representation of the creator's original work to become digital content representing other types of associated IP that can be consumed by an end user. This evolution may increase the value of digital content that, finally, becomes a consumer product.

Several user roles have been identified and they can be part of different value chains, but here we will concentrate on the content creation model and the minimum and necessary user roles required.

Figure 68 represents the relationship between the user roles and the content lifecycle, from the original work in the mind of the creator to the content consumed by the final user. User roles are represented by rounded squares, which contain the role name together with the different content type they can create and register. The label in each arrow describes the type of content that relates two user roles. The user roles appearing in Figure 68 involve not only those in the creation but also those in the distribution value chain.

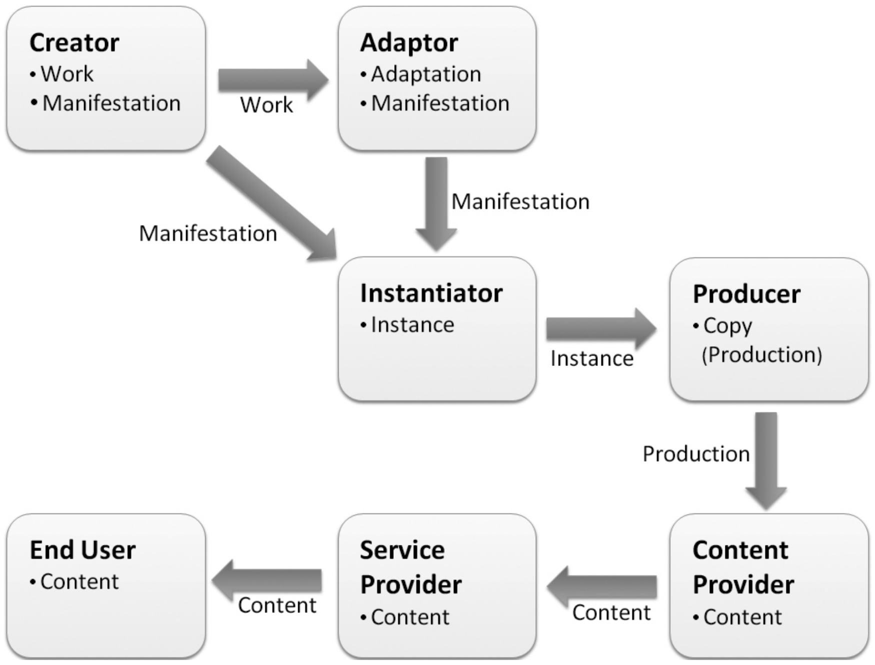


Figure 68. Content lifecycle and user roles.

Table 18 summarises the user roles involved in the creation of content and provides their description, whereas Table 19 describes the evolution of content during its lifecycle.

Table 18. User roles involved in the creation.

User Roles	**Description**
Creator	User role that creates the first manifestation of an original work, not derived from any other work
Adaptor	User role that creates derivative works called adaptations. Users taking this role also create manifestations over their own adaptations
Instantiator	User role that can create instances based on manifestations of original works or adaptations
Producer	User role that can create copies from instances in order to distribute them

Table 19. Content evolution during lifecycle.

Content type	**Description**
Work	A creation that retains intellectual or artistic attributes, i.e. the underlying concept of an artistic work (a song, a play, etc.). It defines the common core that identifies the physical representation of a Work
Adaptation	The modification of an original work made by an adaptor and authorised by the creator (or the corresponding rights holder)
Manifestation	The physical representation of an original work or an adaptation subject to representation in digital form. Depending on the kind of work it may take on many different forms, an example being a recorded song (in digital or analogue format), a manuscript, a music score, etc
Instance	A stylised expression of a manifestation such as a performance of a score that may or may not be on a support such as a media file.
Copy	A content item available to other users as a commercial product

IP Entities defined in the DMP creation model represent the different stages of content in the analogue world, not only taking into account if the IP Entity has a digital/physical representation. To be able to represent and control the actions performed over the different IP Entities by user roles during the content lifecycle, a digital representation of each entity is needed. Thus, apart from the original IP Entities defined in DMP, different DCI representations were required to be defined for the implementation in Linked-Work. Their definition also responds to the need for distinguishing the applicable IP rights to each IP Entity. These DCI representations are summarised next:

- Generic Work. It only contains the metadata such as the creator, title, representative, etc.
- Licensed Work. It includes the Work metadata plus the potential rights and reporting information.
- Work Manifestation. It includes all the information in the Licensed Work plus the description of the physical representation of the work, whether it is analogue or digital and any existing identifiers such as ISWC, ISAN, etc.

- Work Manifestation Copy. It includes all the information in the Work Manifestation plus an associated digital resource. It is not devised to be commercialised but to disseminate the work manifestation by means of the digital resource.
- Adaptation. It includes the metadata, rights and reporting information of the object derived from a licensed work.
- Adaptation Manifestation. It includes all the information in the Adaptation plus the description of the physical representation of the adaptation, whether it is analogue or digital and any existing identifiers such as ISWC, ISAN, etc.
- Adaptation Manifestation Copy. It includes all the information in the Adaptation Manifestation plus an associated digital resource. It is not devised to be commercialised but to disseminate the Adaptation Manifestation by means of the digital resource.
- Instance. It includes the corresponding metadata, potential rights and reporting information and a resource that represents a particular rendition or interpretation of the Work or Adaptation Manifestation Copy resource (e.g. music piece played by a different interpreter). The instance can be commercialised.
- Copy. It is an exact copy of an Instance that is registered by a different actor in order to be distributed and commercialised by the new actor.

Content managed by the different user roles in the value network holds Intellectual Property (IP) rights. As we described in the previous section, in the DMP context, content which holds IP rights represents one or other IP Entity.

The rights that can be associated to the different IP Entities are defined by DMP's Represent Rights Data (RRD) [47]. RRD is the digital representation of the RRD ontology of IP Entities and associated actions taken upon them as well as those taken on their corresponding digital representations as described in the DMP creation model. RRD defines the relationship between IP Entities present in the creation model with user roles and actions. For the formalisation of this ontology, the Ontology Web Language (OWL) is used.

In RRD, actions refer to both those that may be applied over digital objects as well as those that may not. The result of some actions may imply the creation of a new IP Entity. This is the case, for example, of the action MakeAdaptation, which generates a new IP Entity called Adaptation. On the other hand, other actions do not suppose the creation of a new IP Entity. This is the case, for example, of the action Play, which grants permission for the rendering of the IP Entity. For a complete list of actions refer to [70]. The main actions defined in RRD and considered in Linked-Work are described next.

- Make Adaptation. The action of making an Adaptation.
- Make Instance. The action of making an Instance from a Manifestation.
- Make Copy. The Function by which Device A Stores Content in Device B, preserving the original Content in Device A.
- Distribute. The Right to sell, rent and lend.
- Modify Copy. Action of modifying a copy.
- Embed. The Right to include a resource in another resource.

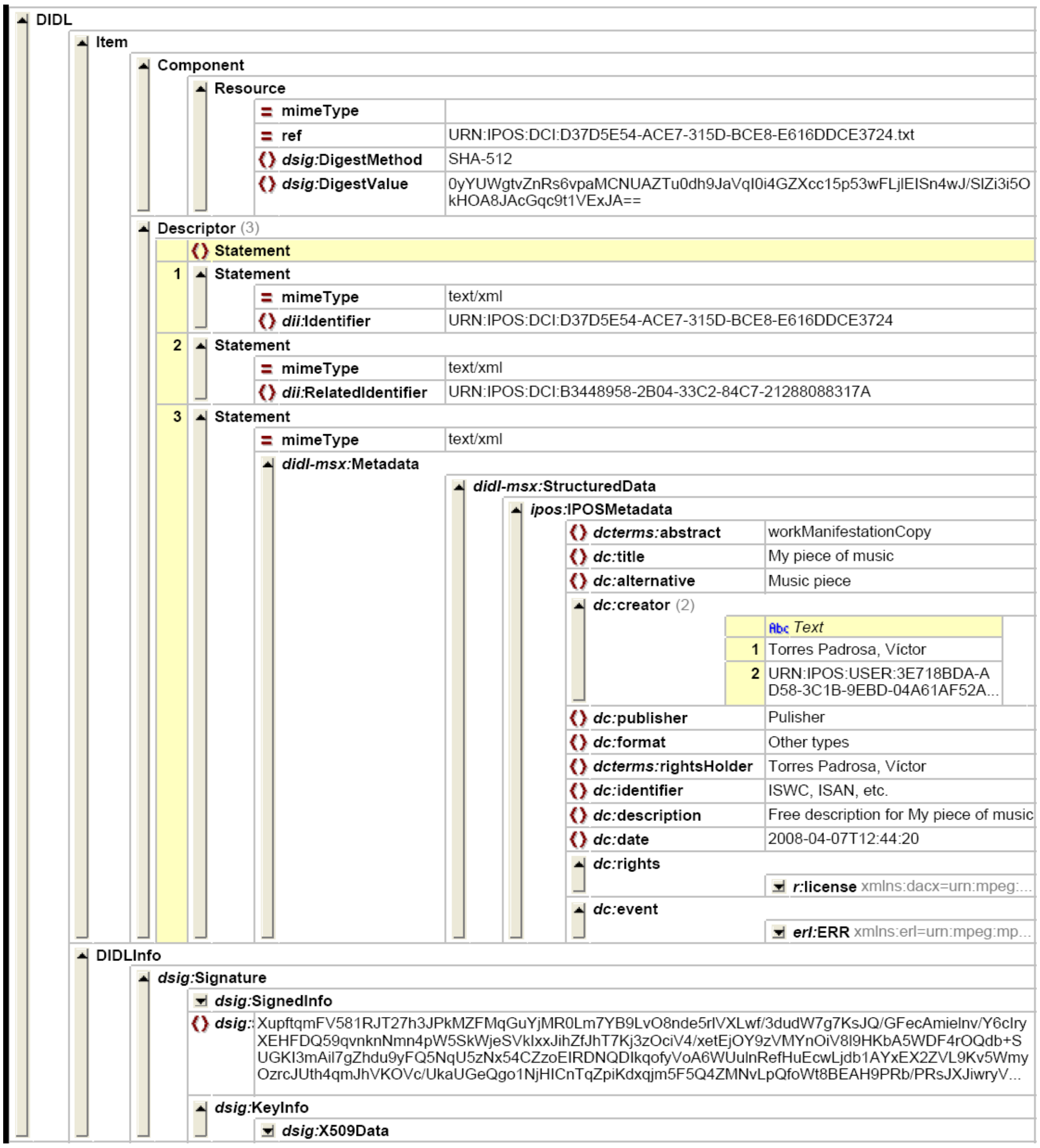
- Enlarge. The Right to modify a resource by making it larger.
- Extract. The Right to derive a new resource by taking a fragment out of an existing resource.
- Modify. The Right to make and save changes to a resource without creating a new resource.
- Reduce. The right to modify a resource by taking away from it.
- Produce. The Function of making Products.
- Render. The Function of generating a human-perceivable signal from a Resource. Includes the case of executing, installing, playing, printing and uninstalling Resources.
- Public Communication. The Function of publicly displaying/performing, e.g. live performance, radio, television, internet streaming, multicast of Instances and Manifestations, and download.
- Synchronisation. Concurrent performance/display of two distinct Works or Adaptation Instances each for a different sense e.g. text and audio or video and song.

6.4.2 Content Representation and Protection

The content representation adopted to represent objects, which involve the metadata and resource, is the Digital Media Project (DMP) Content Information (DCI) defined in the DMP Interoperable DRM Platform 3 (IDP3) [47]. This format is based on the MPEG-21 Digital Item Declaration (DID) [3], which consists in a XML structured language used to include not only the object's metadata but also the content itself.

In particular, in Linked-Work, some specifics have been taken into account:

- It includes Dublin Core (DC) [72] terms to express most of the metadata fields of the object
    - dcterms:abstract. It describes the type of object (e.g. genericWork. licensedWork, workManifestation, etc.)
    - dc:title. It includes the title of the object.
    - dc:alternative. It includes an alternative name for the object. It is also called the "Also Known As field".
    - dc:creator. It includes the name and surname of the object's Author.
    - dc:creator. It includes the unique identifier of the object's Author.
    - dc:publisher. It includes the publisher of the object, if any.
    - dc:format. It describes the format of the object.
    - dcterms:rightsHolder. It states who the rightsHolder is.
    - dc:date. It holds the registration date.
    - dc:rights. It contains the potential rights over the object defined or restricted by the author.
    - dc:event. It holds the Event Report Requests that apply to the object.
    - It uses the DMP <DIDLInfo> and <Signature> tags to include the object's XML Digital Signature in order to provide integrity and authenticity to the registered objects



```
DIDL
  Item
    Component
      Resource
        mimeType
        ref                      URN:IPOS:DCI:D37D5E54-ACE7-315D-BCE8-E616DDCE3724.txt
        dsig:DigestMethod        SHA-512
        dsig:DigestValue         0yYUWgtvZnRs6vpaMCNUAZTu0dh9JaVqI0i4GZXcc15p53wFLjlEISn4wJ/SlZi3i5O
                                 kHOA8JAcGqc9t1VExJA==
    Descriptor (3)
      Statement
    1 Statement
        mimeType                 text/xml
        dii:Identifier           URN:IPOS:DCI:D37D5E54-ACE7-315D-BCE8-E616DDCE3724
    2 Statement
        mimeType                 text/xml
        dii:RelatedIdentifier    URN:IPOS:DCI:B3448958-2B04-33C2-84C7-21288088317A
    3 Statement
        mimeType                 text/xml
        didl-msx:Metadata
          didl-msx:StructuredData
            ipos:IPOSMetadata
              dcterms:abstract        workManifestationCopy
              dc:title                My piece of music
              dc:alternative          Music piece
              dc:creator (2)
                                      Text
                                   1  Torres Padrosa, Víctor
                                   2  URN:IPOS:USER:3E718BDA-A
                                      D58-3C1B-9EBD-04A61AF52A...
              dc:publisher            Pulisher
              dc:format               Other types
              dcterms:rightsHolder    Torres Padrosa, Víctor
              dc:identifier           ISWC, ISAN, etc.
              dc:description          Free description for My piece of music
              dc:date                 2008-04-07T12:44:20
              dc:rights
                                      r:license xmlns:dacx=urn:mpeg:...
              dc:event
                                      erl:ERR xmlns:erl=urn:mpeg:mp...
  DIDLInfo
    dsig:Signature
      dsig:SignedInfo
      dsig::  XupftqmFV581RJT27h3JPkMZFMqGuYjMR0Lm7YB9LvO8nde5rlVXLwf/3dudW7g7KsJQ/GFecAmielnv/Y6clry
              XEHFDQ59qvnknNmn4pW5SkWjeSVklxxJihZfJhT7Kj3zOciV4/xetEjOY9zVMYnOiV8l9HKbA5WDF4rOQdb+S
              UGKI3mAil7gZhdu9yFQ5NqU5zNx54CZzoEIRDNQDlkqofyVoA6WUulnRefHuEcwLjdb1AYxEX2ZVL9Kv5Wmy
              OzrcJUth4qmJhVKOVc/UkaUGeQgo1NjHICnTqZpiKdxqjm5F5Q4ZMNvLpQfoWt8BEAH9PRb/PRsJXJiwryV...
      dsig:KeyInfo
        dsig:X509Data
```

Figure 69. Sample Object Representation.

- It uses the DMP and MPEG-21 <dii:Identifier> tag to enclose the object’s unique identifier
- It uses the DMP and MPEG-21 <dii:RelatedIdentifier> tag to refer to the object’s ancestor.
- It uses the DMP and MPEG-21 <Resource> tag to include information about the Resource, but not the resource itself
    - It uses the ref attribute to refer to the resource name. The location is assumed to be the Content Registration service.
    - It uses the <dsig:DigestValue> tag to include an estimation of the Resource Hash, which can be used to determine whether it is the original Resource or not after being downloaded by any authorised application.

   - It uses the <dsig:DigestMethod> tag to express the Digest Method that has been used to estimate a Hash of the resource, which is SHA-512.

- The resource is not embedded in the object in order to ease the distribution of the objects. The object, by means of the <Resource> tag, contains the reference to the service where it can be retrieved from the Content Registration service. The content is stored in this server encrypted with an asymmetric AES encryption of 128 bytes of block and 128 bytes of key length.

6.4.3 Link between original and derived works

This feature consists in keeping a link between any object representation, which is represented in a digital format by a digital object (also called object representation), and all the works that derive from it. The link is embedded in the object representation, enabling a direct way to discover the ancestor, and recursively, the whole lineage of a given object representation.

Figure 70 presents an OR excerpt, where the solution adopted to implement this feature is presented. The MPEG-21 <dii:Identifier> and <dii:RelatedIdentifier> tags are used for this purpose.

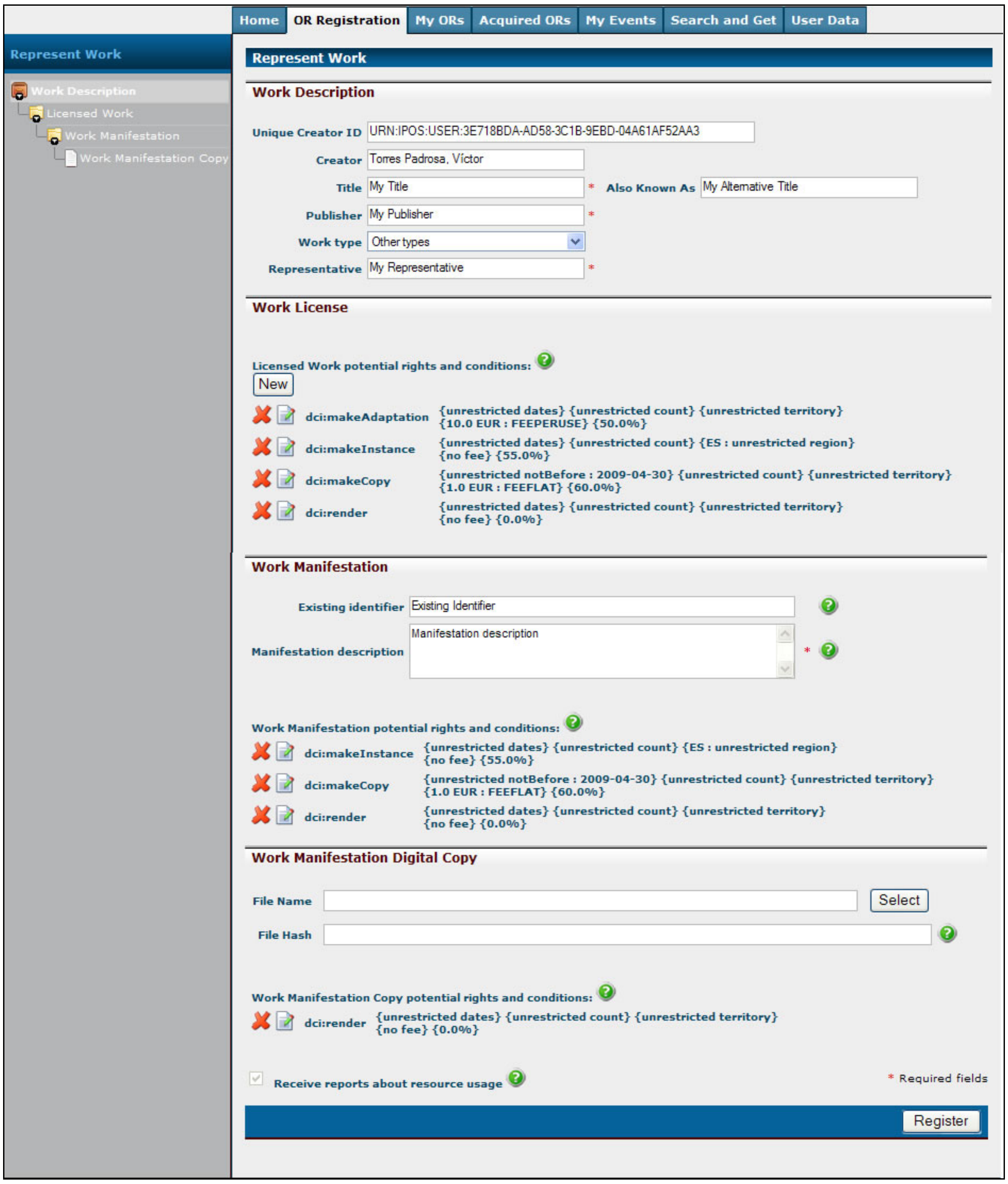
```
<DIDL>
  <Item>
    <Component>
      <Resource ref="URN:LW:DCI:D37D5E54-ACE7-315D-….txt">
        <dsig:DigestMethod>SHA-512</dsig:DigestMethod>
        <dsig:DigestValue>0yYUWgtvZnRs6vpaMCNUAZTu0dh9JaVw...==</dsig:DigestValue>
      </Resource>
    </Component>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dii:Identifier>URN:LW:DCI:D37D5E54-ACE7-315D-…</dii:Identifier>
      </Statement>
    </Descriptor>
    <Descriptor>
      <Statement mimeType="text/xml">
        <dii:RelatedIdentifier>URN:LW:DCI:B3448958-...</dii:RelatedIdentifier>
      </Statement>
    </Descriptor>
  </Item>
</DIDL>
```

Figure 70. Object representation (OR) excerpt – OR linkage.

6.4.4 Potential Rights Definition and Transmission

Potential rights refer to the potential rights and conditions associated to an object, which correspond to those the object's author would be willing to grant to any user, including the object in which they are included or any object which derives from it along the content value chain. For example, if a Creator assigns the rights MakeAdaptation and PublicCommunication to a Licensed Work, it means that potentially, Adaptations can be created over the Licensed Work and PublicCommunication can be performed over the Instances or Copies derived from the object. The term "potentially" means that the author is willing to give those rights under some specific conditions. However, those rights under those conditions first need to be acquired by the user who wants to exploit them, before they can be exercised by that specific user. Any user will not be able to acquire any right or condition which is not included in the object's potential rights, unless a new object with new potential rights is created by the author for their own interest.

In Linked-Work, potential rights are defined by the content Creator when registering the Licensed Work. Those potential rights need to cover any of the actions that may be performed when deriving any object from the Creator's object, including Adaptations, Instances and Copies.



Represent Work

Work Description

Unique Creator ID	URN:IPOS:USER:3E718BDA-AD58-3C1B-9EBD-04A61AF52AA3
Creator	Torres Padrosa, Víctor
Title	My Title *
Also Known As	My Alternative Title
Publisher	My Publisher *
Work type	Other types
Representative	My Representative *

Work License

Licensed Work potential rights and conditions:

New

dci:makeAdaptation	{unrestricted dates} {unrestricted count} {unrestricted territory} {10.0 EUR : FEEPERUSE} {50.0%}
dci:makeInstance	{unrestricted dates} {unrestricted count} {ES : unrestricted region} {no fee} {55.0%}
dci:makeCopy	{unrestricted notBefore : 2009-04-30} {unrestricted count} {unrestricted territory} {1.0 EUR : FEEFLAT} {60.0%}
dci:render	{unrestricted dates} {unrestricted count} {unrestricted territory} {no fee} {0.0%}

Work Manifestation

Existing identifier	Existing Identifier
Manifestation description	Manifestation description *

Work Manifestation potential rights and conditions:

dci:makeInstance	{unrestricted dates} {unrestricted count} {ES : unrestricted region} {no fee} {55.0%}
dci:makeCopy	{unrestricted notBefore : 2009-04-30} {unrestricted count} {unrestricted territory} {1.0 EUR : FEEFLAT} {60.0%}
dci:render	{unrestricted dates} {unrestricted count} {unrestricted territory} {no fee} {0.0%}

Work Manifestation Digital Copy

File Name	
File Hash	

Select

Work Manifestation Copy potential rights and conditions:

dci:render	{unrestricted dates} {unrestricted count} {unrestricted territory} {no fee} {0.0%}

✓ Receive reports about resource usage

* Required fields

Register

Figure 71. Example of transmission when editing the Licensed Work potential rights without restricting the conditions in the Work Manifestation and Work Manifestation Copy.

Whenever an object with potential rights is registered, the License service generates a license template for each of the rights plus conditions that may be acquired for that object. This template is used to offer the rights and conditions for being acquired. Once acquired, a specific license is generated for the specific user that acquires it.

When a derived object is created, it inherits the potential rights from its ancestor, if any. In this way, the rights and conditions in the derived object are equal to those in the ancestor. However, the potential rights inside the derived object may be restricted in terms of rights and even conditions. This means that the derived object may not include some of the rights in the ancestor, may restrict the right scope according to the rights hierarchy defined in the RDD or may even restrict the conditions which apply to the object. In the previous example, the Manifestation potential rights may only involve rights such as PublicCommunication or some of the rights which hierarchically derive from it, such as Broadcast, Download or Stream. The potential rights may be also restricted in terms of temporal, usage, territory or rights over derivative conditions.

Next we detail the conditions that can not be freely modified when creating a derived object:

- Not Before: it cannot be modified by a previous date
- Not After: it cannot be modified by a posterior date
- Usage limit: if present, it cannot be increased
- Country: if present, it cannot be modified
- Region: if present, it cannot be modified
- Economical conditions
    - Amount: if present, it cannot be decreased
    - Currency: if present, it cannot be modified
    - Fee Type: if present, it cannot be modified
- Rights Transferred over derivative: if present, it cannot be increased

6.4.5 Event Reporting Transmission

In this section we analyse how Event Reporting information can be transmitted across the content value chain so that Event Reports are generated not only towards the object author but also towards all the authors of the object’s ancestors.

Every time a user registers a new object in the system, they are asked whether they want to receive reports about the usage of their objects in the system. For example, if an author registers a LicensedWork with the potential right MakeAdaptation, it means that whenever a user performs the action MakeAdaptation, a usage report will be generated towards the original author.

Linked-Work implements the reporting functionality making use of MPEG-21 Event Reporting standard by making use of Event Report Requests (ERRs). Event Report requests are included in the registered objects as a means of publicly stating the cases when Event Reports will be generated. However, in order to ensure the transmission of the request along the whole content value chain, a customised and enhanced approach has been tested and adopted.

ERRs are transmitted from one object to its derivatives by a similar process as that described in previous section. This means that an object will have not only the ERRs belonging to the author of the object, but also those coming from the object’s ancestors.

In this way, the execution of an operation over an object may unleash the generation of several Event Reports, each of them directed to a specific user that corresponds to the author of some of the object's ancestors. The web application and Certification service are responsible for processing the object so that it includes the inherited information.

Figure 72 provides a graphical example on how Event Report Requests are transmitted from an object towards its derivatives and how an action unleashes the generation of an Event Report towards all the actors in the content value chain.

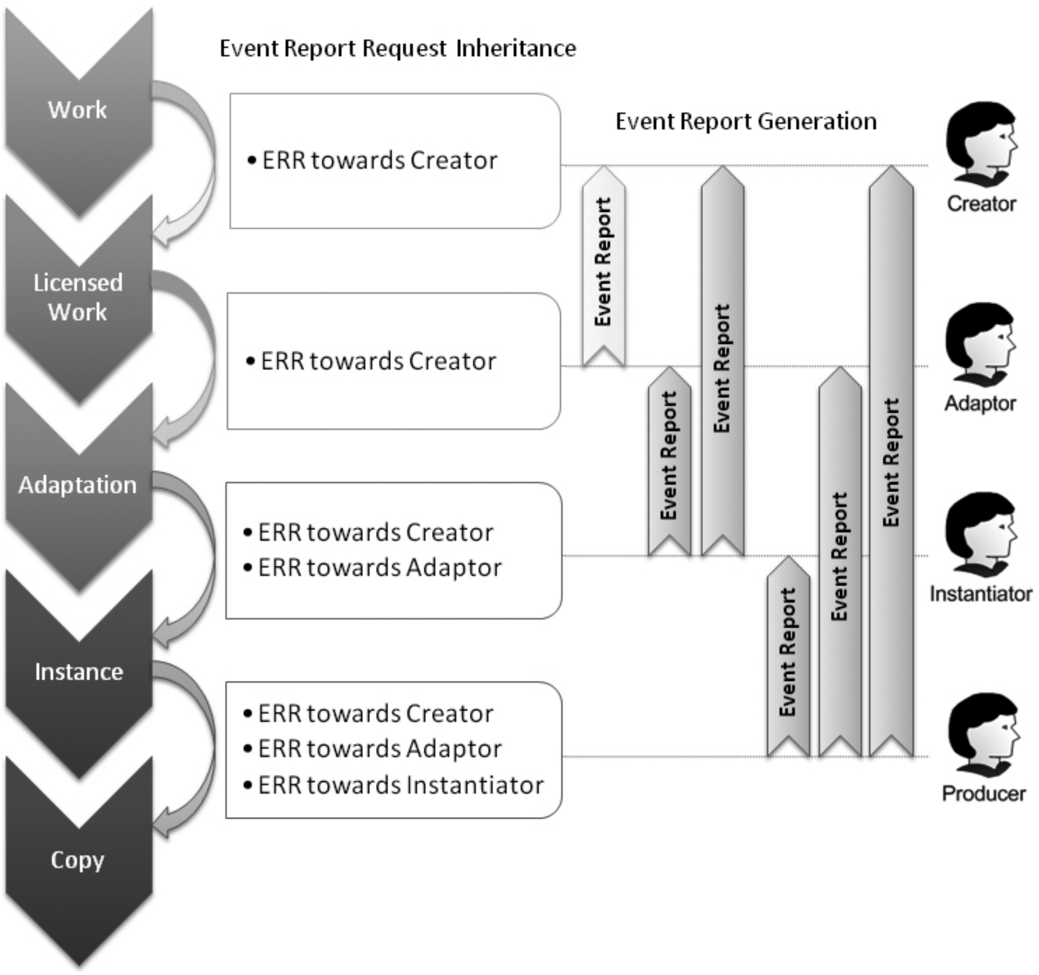


Figure 72. ERR inheritance and ER generation in Linked-Work.

Figure 73 and Figure 74 provide an example of an original ERR included in an object and an inherited ERR in the same object. Note that the recipient is different in both, as the original ERR is directed to the object's author, whereas the inherited ERR is directed to the author of the object's ancestor from which the inherited ERR comes.

```
<erl:ERR xmlns:erl="urn:mpeg:mpeg21:2005:01-ERL-NS"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004" xmlns:ns="http://www.w3.org/2001/XMLSchema-
instance" xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
ns:schemaLocation="urn:mpeg:mpeg21:2002:01-DII-NS ../schemas/DMP_dii.xsd
urn:mpeg:mpeg21:2003:01-REL-R-NS ../schemas/rel-r-dmpREL.xsd urn:mpeg:mpeg21:2005:01-ERL-
NS ../schemas/erl.xsd urn:mpeg:mpeg7:schema:2004 ../schemas/mpeg7.xsd">
        <erl:ERRDescriptor>
                <erl:Modification>
                        <erl:UserId>URN:LW:USER:FEE4704D-449B-3D49-A223-
E2CFBC638A3F</erl:UserId>
```

```
            <erl:Description>Linked-Work Event Report Request</erl:Description>
        </erl:Modification>
    </erl:ERRDescriptor>
    <erl:ERSpecification>
        <erl:ERDescription>Linked-Work Event Report</erl:ERDescription>
        <erl:ERPayloadSpecification>
            <erl:UserId/>
            <erl:Location/>
            <erl:Operation/>
            <erl:Time/>
            <erl:DomainData reportTag="LicenseId" syntax="xsd:string"/>
            <erl:DomainData reportTag="AuthResult" syntax="xsd:string"/>
            <erl:DomainData reportTag="DerivedDCIId" syntax="xsd:string"/>
            <erl:DIMetadata>
                <erl:DIMetadataElement tagName="abstract"/>
                <erl:DIMetadataElement tagName="title"/>
                <erl:DIMetadataElement tagName="alternative"/>
                <erl:DIMetadataElement tagName="creator"/>
                <erl:DIMetadataElement tagName="creator"/>
                <erl:DIMetadataElement tagName="identifier"/>
                <erl:DIMetadataElement tagName="date"/>
            </erl:DIMetadata>
        </erl:ERPayloadSpecification>
        <erl:ERDeliverySpecification>
            <erl:Recipient>
                <erl:UserId>URN:LW:USER:FEE4704D-449B-3D49-A223-
E2CFBC638A3F</erl:UserId>
            </erl:Recipient>
        </erl:ERDeliverySpecification>
    </erl:ERSpecification>
    <erl:EventConditionDescriptor>
        <erl:Operator name="("/>
        <erl:DIOperationCondition>
            <erl:DIOperationEvent>
                <erl:Operation>dci:makeManifestationCopy</erl:Operation>
                <erl:DII>URN:LW:DCI:1D3F4E75-7BF4-3DA5-90B3-
688E56F90337</erl:DII>
            </erl:DIOperationEvent>
        </erl:DIOperationCondition>
        <erl:Operator name="OR"/>
        <erl:DIOperationCondition>
            <erl:DIOperationEvent>
                <erl:Operation>dci:makeInstance</erl:Operation>
                <erl:DII>URN:LW:DCI:1D3F4E75-7BF4-3DA5-90B3-
688E56F90337</erl:DII>
            </erl:DIOperationEvent>
        </erl:DIOperationCondition>
        <erl:Operator name=")"/>
        <erl:Operator name="OR"/>
        <erl:Operator name="("/>
        <erl:DIOperationCondition>
            <erl:DIOperationEvent>
                <erl:Operation>dci:makeCopy</erl:Operation>
                <erl:DII>URN:LW:DCI:1D3F4E75-7BF4-3DA5-90B3-
688E56F90337</erl:DII>
            </erl:DIOperationEvent>
        </erl:DIOperationCondition>
        <erl:Operator name="OR"/>
        <erl:DIOperationCondition>
            <erl:DIOperationEvent>
```

```
                                        <erl:Operation>dci:render</erl:Operation>
                                        <erl:DII>URN:LW:DCI:1D3F4E75-7BF4-3DA5-90B3-
688E56F90337</erl:DII>
                              </erl:DIOperationEvent>
                    </erl:DIOperationCondition>
                    <erl:Operator name=")"/>
          </erl:EventConditionDescriptor>
</erl:ERR>
```

Figure 73. Sample Event Report Request present in an Object.

```
<erl:ERR xmlns:erl="urn:mpeg:mpeg21:2005:01-ERL-NS"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2004" xmlns:ns="http://www.w3.org/2001/XMLSchema-
instance" xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS"
ns:schemaLocation="urn:mpeg:mpeg21:2002:01-DII-NS ../schemas/DMP_dii.xsd
urn:mpeg:mpeg21:2003:01-REL-R-NS ../schemas/rel-r-dmpREL.xsd urn:mpeg:mpeg21:2005:01-ERL-
NS ../schemas/erl.xsd urn:mpeg:mpeg7:schema:2004 ../schemas/mpeg7.xsd">
          <erl:ERRDescriptor>
                    <erl:Modification>
                              <erl:UserId>URN:LW:USER:3E718BDA-AD58-3C1B-9EBD-
04A61AF52AA3</erl:UserId>
                              <erl:Description>Inherited Linked-Work  Report Request</erl:Description>
                    </erl:Modification>
          </erl:ERRDescriptor>
          <erl:ERSpecification>
                    <erl:ERDescription>Inherited Linked-Work Event Report</erl:ERDescription>
                    <erl:ERPayloadSpecification>
                              <erl:UserId/>
                              <erl:Location/>
                              <erl:Operation/>
                              <erl:Time/>
                              <erl:DomainData reportTag="LicenseId" syntax="xsd:string"/>
                              <erl:DomainData reportTag="AuthResult" syntax="xsd:string"/>
                              <erl:DomainData reportTag="DerivedDCIId" syntax="xsd:string"/>
                              <erl:DIMetadata>
                                        <erl:DIMetadataElement tagName="abstract"/>
                                        <erl:DIMetadataElement tagName="title"/>
                                        <erl:DIMetadataElement tagName="alternative"/>
                                        <erl:DIMetadataElement tagName="creator"/>
                                        <erl:DIMetadataElement tagName="creator"/>
                                        <erl:DIMetadataElement tagName="identifier"/>
                                        <erl:DIMetadataElement tagName="date"/>
                              </erl:DIMetadata>
                    </erl:ERPayloadSpecification>
                    <erl:ERDeliverySpecification>
                              <erl:Recipient>
                                        <erl:UserId>URN:LW:USER:3E718BDA-AD58-3C1B-9EBD-
04A61AF52AA3</erl:UserId>
                              </erl:Recipient>
                    </erl:ERDeliverySpecification>
          </erl:ERSpecification>
          <erl:EventConditionDescriptor>
                    <erl:Operator name="("/>
                    <erl:DIOperationCondition>
                              <erl:DIOperationEvent>
                                        <erl:Operation>dci:makeManifestation</erl:Operation>
                                        <erl:DII>URN:LW:DCI:1D3F4E75-7BF4-3DA5-90B3-
688E56F90337</erl:DII>
                              </erl:DIOperationEvent>
```

```
                </erl:DIOperationCondition>
                <erl:Operator name="OR"/>
                <erl:DIOperationCondition>
                        <erl:DIOperationEvent>
                                <erl:Operation>dci:makeManifestationCopy</erl:Operation>
                                <erl:DII>URN:LW:DCI:1D3F4E75-7BF4-3DA5-90B3-
688E56F90337</erl:DII>
                        </erl:DIOperationEvent>
                </erl:DIOperationCondition>
                <erl:Operator name=")"/>
                <erl:Operator name="OR"/>
                <erl:Operator name="("/>
                <erl:DIOperationCondition>
                        <erl:DIOperationEvent>
                                <erl:Operation>dci:makeAdaptation</erl:Operation>
                                <erl:DII>URN:LW:DCI:1D3F4E75-7BF4-3DA5-90B3-
688E56F90337</erl:DII>
                        </erl:DIOperationEvent>
                </erl:DIOperationCondition>
                <erl:Operator name="OR"/>
                <erl:DIOperationCondition>
                        <erl:DIOperationEvent>
                                <erl:Operation>dci:makeInstance</erl:Operation>
                                <erl:DII>URN:LW:DCI:1D3F4E75-7BF4-3DA5-90B3-
688E56F90337</erl:DII>
                        </erl:DIOperationEvent>
                </erl:DIOperationCondition>
                <erl:Operator name=")"/>
                <erl:Operator name="OR"/>
                <erl:Operator name="("/>
                <erl:DIOperationCondition>
                        <erl:DIOperationEvent>
                                <erl:Operation>dci:makeCopy</erl:Operation>
                                <erl:DII>URN:LW:DCI:1D3F4E75-7BF4-3DA5-90B3-
688E56F90337</erl:DII>
                        </erl:DIOperationEvent>
                </erl:DIOperationCondition>
                <erl:Operator name="OR"/>
                <erl:DIOperationCondition>
                        <erl:DIOperationEvent>
                                <erl:Operation>dci:render</erl:Operation>
                                <erl:DII>URN:LW:DCI:1D3F4E75-7BF4-3DA5-90B3-
688E56F90337</erl:DII>
                        </erl:DIOperationEvent>
                </erl:DIOperationCondition>
                <erl:Operator name=")"/>
        </erl:EventConditionDescriptor>
</erl:ERR>
```

Figure 74. Sample Inherited Event Report Request present in the same Object as previous figure.

6.4.6 Rights over Derivative Objects

In this section, we analyse how to include inside the definition of the potential rights a condition to determine the percentage of rights that are transferred towards the derivatives of an object or, in other words, the percentage of rights that the object's author preserves over any object derived from theirs.

For example, an author may want to enable the registration of Adaptations derived from their LicensedWork, but only if the author keeps a percentage of the rights over the Adaptation and successive objects in the lineage.

This restriction can be seen as an additional condition associated to the right that may be granted. In the example, we could say: The user U may MakeAdaptation over the object O under the condition “the author preserves a certain percentage of the rights over the derivatives of the resulting object”.

This condition is slightly different from common MPEG-21 Rights Expression Language (REL) [4] or even the Open Mobile Alliance Digital Rights Management [7] Rights Expression Language (REL) [43] conditions, as it applies over the derivatives of an object instead of over the object itself. Thus, this condition cannot be taken into account when authorising the creation of derived objects, but when authorising the creation of objects derived from the derived, i.e. a second level of derivation.

The solution adopted to deal with this new condition has consisted on extending the MPEG-21 REL, used to express potential rights and licenses in Linked-Work, in order to add a new condition, which expresses the percentage of rights which are transferred. Thus, the percentage of rights preserved could be determined as:

100 – (percentage of rights which are transferred)

Figure 75. Computation of the percentage of preserved rights.

The transferred rights over derivatives are expressed in the following form:

```
<lw:rightsOverDerivative>
        <lw:percentage>50.0</lw:percentage>
</lw:rightsOverDerivative>
```

Figure 76. Expression of the rights over derivative condition in XML.

The fact of preserving a percentage of the rights over the derivative objects implies that whenever a payment is cleared, corresponding to the amount expressed in the license terms, if the licenses in the object’s value chain specify any rightsOverDerivative condition, the fee needs to be distributed amongst all the authors involved in the process.

6.4.7 Incomes distribution

The distribution of the incomes is determined by the author that registers the Work. Following the example provided in previous sections, George, who is a creator, by setting the conditions of the rights that apply over the creation value chain is determining the minimum incomes he will perceive. Let’s see how it works in two examples.

In the first example George determines in his Work:

- Make Adaptation, with a per use fee condition of 10 EUR. Transferred Rights Over Derivatives are set to **80**%.
- Make Instance, with a per use fee condition of 100 EUR. Transferred Rights Over Derivatives are set to **85**%.

- MakeCopy, with a per use fee condition of **0,5** EUR. Transferred Rights Over Derivatives are set to 100%.

This means that George will perceive, a minimum amount of (1- **0,8**) · (1 - **0,85**) · **0,5** EUR for each Copy registered in the system, i.e. 0,015 EUR per Copy.

In the second example George determines:

- Make Adaptation, with a per use fee condition of 10 EUR. Transferred Rights Over Derivatives are set to 20%.
- Make Instance, with a per use fee condition of 100 EUR. Transferred Rights Over Derivatives are set to 25%.
- MakeCopy, with a per use fee condition of 1,5 EUR. Transferred Rights Over Derivatives are set to 100%.

This means that George will perceive, a minimum amount of (1- **0,2**) · (1 - **0,25**) · **1,5** EUR for each Copy registered in the system, i.e. 0,9 EUR per Copy.

The conditions set by each author will depend on their personal preferences and also on what the market is willing to pay for it.

6.5 Use Cases

6.5.1 Work Registration

George, who is a Creator, has recently created a new work. One day he decides to register his creation by using Linked-Work applications so that he will be able to send copies of it to some potential clients or friends that may want to distribute, adapt or instance it.

George needs to be a registered user of the Linked-Work system. Once registered, users are able to use Linked-Work applications: Linked-Work web application and Linked-Work player.

George will use the web application to register the metadata describing his Work (Generic Work), together with the potential rights that will be applicable (Licensed Work), after a payment agreement, over the Work to create derivative Adaptations, Instances or Copies. The first Manifestation of the Work (Work Manifestation) will be also registered automatically together with the Generic Work and Licensed Work. Optionally, a digital Copy of the Manifestation (Work Manifestation Copy) may be also registered in that moment.

George selects the potential rights and conditions that will be offered to other users of the system, offering the following:

- Make Adaptation, with a per use fee condition of 10 EUR, and a time condition which states that the adaptation must be made before 1 year, providing a limit date. Transferred Rights Over Derivatives are set to 80%.
- Make Adaptation, with a flat fee condition of 50 EUR and a count limit of 10 times, and a time condition which states that the adaptation must be made before 1 year, providing a limit date. Transferred Rights Over Derivatives are set to 80%.
- Make Instance, with a per use fee condition of 100 EUR, and a time condition which states that the adaptation must be made before 1 year, providing a limit date. Transferred Rights Over Derivatives are set to 85%.

- MakeCopy, with a per use fee condition of 0,5 EUR, and a time condition which states that the adaptation must be made before 1 year, providing a limit date. Transferred Rights Over Derivatives are set to 100%.
- PublicCommunication, with a per use fee condition of 100 EUR, and a time condition which states that the adaptation must be made before 6 months, providing a limit date. Transferred Rights Over Derivatives are set to 100%.
- Render, without any restrictions or fees. Transferred Rights Over Derivatives are set to 0%.

Once registered, George will be able to send or distribute the registered Work Manifestation Copy DCI amongst his friends or potential clients to communicate the Work to Adaptors, Instantiators and other value chain users further down the chain. The Work Manifestation Copy object will be useful to make diffusion of George’s Work Manifestation and the rights that can be exercised over it are restricted to a single right: the Render right. In this way, the Work Manifestation Copy can be only used to spread George’s Work.

6.5.2 Work diffusion by means of the Work Manifestation Copy

Following the example provided in the previous section, once George has registered his Work, Licensed Work, Work Manifestation and Work Manifestation Copy, he is able to spread the registered Work Manifestation Copy DCI amongst his friends or potential clients. The Work Manifestation Copy (WMC) object can be sent by any means, as e.g. e-mail.

George decides to place his WMC object in his personal website, where he usually provides the latest news about his creations. George’s website refers to Linked-Work website, where users can register and download the Linked-Work Player, which is able to manage those kind of objects that provide access to his new creation.

Paula, who is an important piano player and one of George’s website common visitors, finds out the entry describing George’s new creation and downloads the associated object. Paula also accesses Linked-Work main site, downloads the Player application and opens George’s object in her computer. The player presents the object’s metadata and provides a Link to the main web application, where any user can register into the system. Paula sees there are some operations in the player that are only available for registered users, such as getting a license to view the resource associated to George’s creation and viewing the resource. Thus, Paula decides to register into Linked-Work. She accesses the web portal and provides her personal data through a web form. Next, she receives a confirmation mail where she gets a temporal user name and password which can be used to access the web application and some of the operations in the player. Although she is suggested to change the user and password, she decides to log into the player. After logging in, the player enables Paula to go to the web application section where she can acquire a license for viewing George’s resource. She has only one option, which comes from the different rights and conditions offered by George, that is, acquiring the right for rendering the resource associated to the ManifestationCopy without any restrictions. (Note: other rights such as Make Adaptation and Make Instance do not apply the Work Manifestation Copy). Paula decides to acquire this license, as it is free.

Once acquired, in the player she tries to view the resource so, after the application checks Paula is authorised to render the resource, it downloads the resource, decrypts it and renders it.

6.5.3 Adaptation Registration

Following the example provided in previous section, Paula likes very much George's creation, which consists in a music piece played with the guitar.

Immediately, she takes her piano and begins to play it, while realises some improvements can be made over the piece. Paula realises she can use that piece for her new album, so after working hard for some weeks, she decides to register her new work as an Adaptation of George's. For that purpose, Paula accesses the web application, searches George's WMC and navigates through the ancestors of that object, until she finds the corresponding Licensed Work, for which she can acquire the Make Adaptation right. Once she accesses the acquisition interface, she realises there are two options, which come from George's decision:

- Make Adaptation, with a per use fee condition of 10 EUR, and a time condition which states that the adaptation must be made before 1 year, providing a limit date. Transferred Rights Over Derivatives are set to 80%.
- Make Adaptation, with a flat fee condition of 50 EUR and a count limit of 10 times, and a time condition which states that the adaptation must be made before 1 year, providing a limit date. Transferred Rights Over Derivatives are set to 80%.

Paula only wants to register one adaptation of George's Work, so she decides to choose the first option. She does not clear the payment in the moment, but she acquires a payment engagement, which will be satisfied e.g. every month, when Linked-Work clears all pending fees and payments.

After acquiring the license for making the adaptation, she is automatically redirected to the acquired objects section, where she can register the Adaptation, which derives form the Licensed Work for which she has acquired the license. Before registering the Adaptation, she needs to provide the related metadata (the same as for the Work registration) for the Adaptation, Adaptation Manifestation and optionally an Adaptation Manifestation digital Copy. The potential rights that can be assigned to these objects are restricted to those that were selected by George. This means that Paula cannot add more rights than those that were included by George and that can be applied over the Adaptation. Thus, Paula can only include the rights Make Instance and Render to her Adaptation and Adaptation Manifestation, and only Render to her Adaptation Manifestation Copy (AMC). What she can do is to restrict more the conditions present for the rights, as explained in section 6.4.4. Paula decides to increase some fees and restrict some time conditions:

- Make Instance, with a per use fee condition of 120 EUR, and a time condition which states that the adaptation must be made before 6 months, providing a limit date. Transferred Rights Over Derivatives kept to 85%.
- MakeCopy, with a per use fee condition of 0,5 EUR, and a time condition which states that the adaptation must be made before 1 year, providing a limit date. Transferred Rights Over Derivatives are set to 100%.

- PublicCommunication, with a per use fee condition of 200 EUR, and a time condition which states that the adaptation must be made before 3 months, providing a limit date. Transferred Rights Over Derivatives are kept to 100%.
- Render, without any restrictions or fees. Transferred Rights Over Derivatives are set to 0%.

Once registered, Paula is able to send or distribute the registered AMC object amongst her friends or potential clients. The AMC object will be useful to make diffusion of Paula's Adaptation Manifestation and the rights that can be exercised over it are restricted to a single right: the Render right. In this way, the AMC can be only used to spread Paula's Adaptation.

6.5.4 Instance Registration

Paula realises that some pubs or other kind of shops may be interested on using her music for being rendered. Therefore, she decides to register an Instance of her Adaptation Manifestation, so that anyone interested on it is capable of acquiring a license for it.

Paula accesses the Linked-Work web application and searches amongst her own objects. Paula selects the adaptation described in previous section and selects the operation to register an Instance from it. While registering the Instance, she provides the requested metadata and restricts, if needed, the rights and conditions inherited from the Adaptation, setting the following:

- PublicCommunication, with a per use fee condition of 200 EUR, and a time condition which states that the adaptation must be made before 3 months, providing a limit date. Transferred Rights Over Derivatives are kept to 100%.
- MakeCopy, with a per use fee condition of 0,5 EUR, and a time condition which states that the adaptation must be made before 1 year, providing a limit date. Transferred Rights Over Derivatives are set to 100%.
- Render, without any restrictions or fees. Transferred Rights Over Derivatives are set to 0%.

On the other hand, Víctor, who is an intermediary, has heard about Paula's Adaptation and wants to commercialise it. For that purpose, Víctor acquires a license that enables him to register an Instance from Paula's Adaptation Manifestation. The acquisition involves a payment engagement towards Paula of 120 EUR per Instance registered. However, as the Instance is a derivative of the Paula's Adaptation, Paula keeps the 80% of the incomes, while the 20% are transferred to George. Víctor orders a virtuoso piano player different from Paula to interpret her music piece and registers a digital copy as an Instance. Víctor selects the following conditions for the Instance:

- PublicCommunication, with a per use fee condition of 300 EUR, and a time condition which states that the adaptation must be made before 3 months, providing a limit date. Transferred Rights Over Derivatives are kept to 100%.
- MakeCopy, with a per use fee condition of 0,6 EUR, and a time condition which states that the adaptation must be made before 1 year, providing a limit date. Transferred Rights Over Derivatives are set to 100%.
- Render, without any restrictions or fees. Transferred Rights Over Derivatives are set to 0%.

After this process, there are two Instances of Paula’s Adaptation. Both Paula and Víctor can now use their contacts to spread their own Instance.

Paula will get 200 EUR for any PublicCommunication made from her Instance. However, as these incomes are derived from the Instance, which derives her own Adaptation Manifestation and Adaptation, which on their turn derive from George’s Licensed Work, Paula will have to share those incomes by keeping the 80% of the total amount and transferring the 20% to George.

On the other hand, Víctor will get 300 EUR per PublicCommunication performed over his Instance. However, as those incomes are derived from the Instance that derives Paula’s Adaptation Manifestation and Adaptation, Víctor will have to share those incomes by keeping the 85% and transferring the 15% to Paula. Paula, on her turn, will have to transfer 15% of the previous 15% to George, while preserving 85% of the 15%, as her Adaptation derives from Georges Licensed Work.

6.5.5 Copy Registration

Following the example provided in previous section, a record company is interested on commercialising the virtuoso record of Paula’s Adaptation of George’s Work. Víctor has contacted this company and provided them his Instance. The record company registers a Copy object for each of the copies it wants to trade.

The registration of the Copies involves a payment of 0,6 EUR to Víctor, as the registered copies derive from his Instance. However, those incomes correspond to the derivatives of the Paula’s Adaptation Manifestation, Víctor keeps the 85%, whereas Paula perceives the 15%. Moreover, as Paula’s incomes correspond to the Adaptation and Adaptation Manifestation, which derive from George’s Licensed Work, George perceives a 20% of the 15% and Paula keeps 80% of the 15%.

If the record company distributes 100000 copies:

- The record company will pay 100000 · 0,6 EUR = 60000 EUR to Víctor
- Víctor will keep 0,85 · 60000 EUR = 51000 EUR
- Víctor will transfer 0,15 · 60000 EUR = 9000 EUR to Paula
- Paula will earn 0,8 · 9000 EUR = 7200 EUR
- Paula will transfer 0,2 · 9000 EUR = 1800 EUR to George
- George will earn 1800 EUR

6.5.6 Graphical example

Figure 77 depicts a use case where a part of the value chain is covered to illustrate how the payments should be transferred across the value network and prove the feasibility of proposed models.

In the example, the Creator registers a Work and Licensed Work. No license is needed for this action, as the owner of the Work is the Creator.

The Adaptor, after acquiring License 1 is able to register an Adaptation derived from the Licensed Work. When registering the Adaptation, an Event Report and a payment duty will be generated and registered in the system from the Adaptor towards the

Creator. The amount will be that specified in the license that enables the registration of the Adaptation, i.e. 10$.

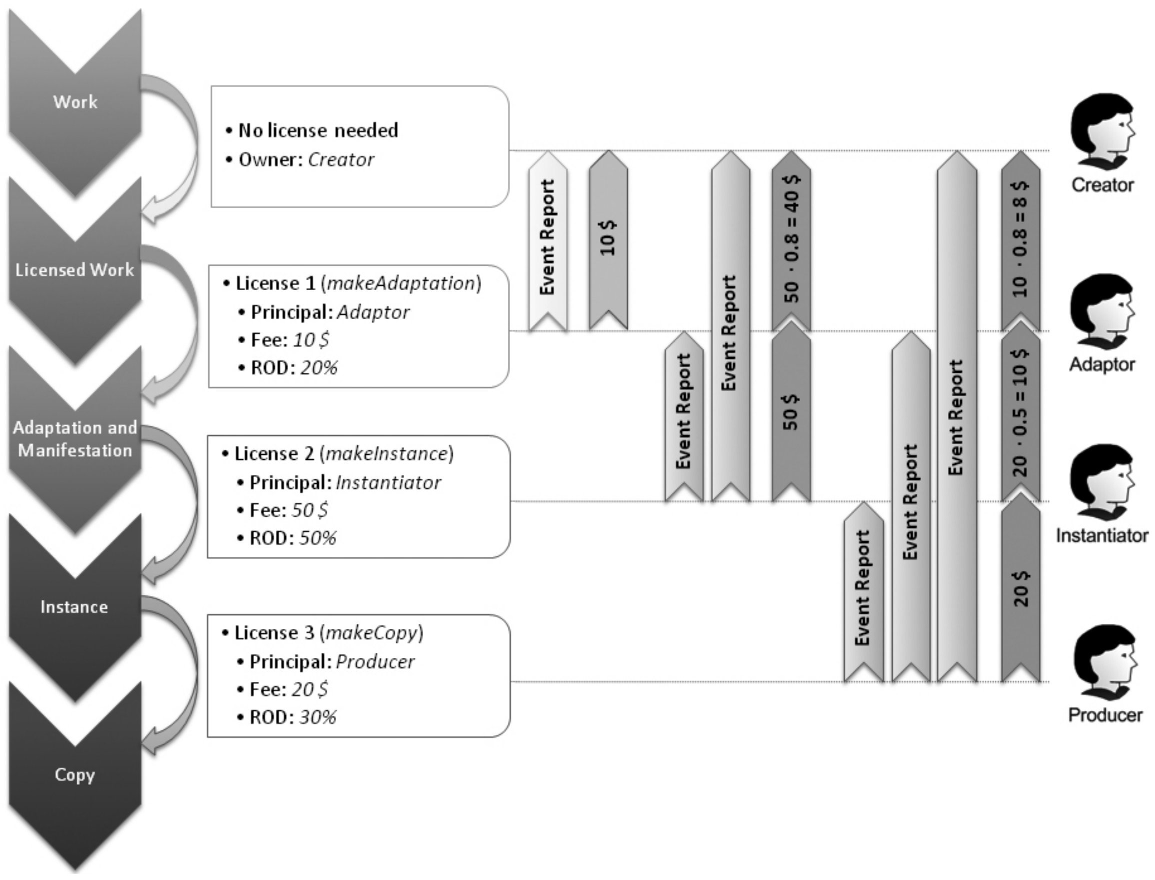


Figure 77. Creation model, user roles, potential rights, event reports and incomes distribution.

The Instantiator, after acquiring License 2, is able to register an Instance derived from previous Adaptation. When registering the Instance, two Event Reports will be generated and registered in the system: one from the Instantiator to the Adaptor and another one from the Instantiator to the Creator, as the Creator also belongs to the content value chain. Regarding the incomes distribution, a payment duty will be registered from the Instantiator to the Adaptor. The amount will be that specified in the license that enables the registration of the Instance, i.e. 50$. Moreover, in this case, another payment duty will be generated from the Adaptor to the Creator. The payment duty will cause the Adaptor's incomes to be shared with the Creator according to the percentage established in License 1. That is, as the Instance is considered a derivative of the Adaptation and the Creator transferred only the 20% of the rights over derivatives in License 1, this means that the Creator preserves the 80% of the incomes coming from any Instance. That's why

Figure 77 depicts that 80% of the 50$ perceived by the Adaptor is transmitted to the Creator, which supposes 40$.

Finally, The Producer, after acquiring License 3, is able to register a Copy derived from previous Instance. When registering the Copy, three Event Reports will be generated and registered in the system: the first from the Producer towards the Instantiator; the second one from the Producer to the Adaptor; and the last one from the Producer to the Creator. That is, an Event Report for each of the actors involved in the content value chain. Regarding the incomes distribution, a payment duty will be registered from the Producer to the Instantiator. The amount will be that specified in the license that enables the registration of the Copy, i.e. 20$. Moreover, in this case, two more payment duties will be generated: one from the Instantiator to the Adaptor and another one from the Adaptor to the Creator. The payment duty will cause the Instantiator's incomes to be shared with the Adaptor and Creator according to the percentage established in Licenses 2 and 1.

As the Copy is considered a derivative from the Instance and the Adaptor transferred only the 50% of the rights over derivatives (ROD) in License 2, this means that the Adaptor preserves the 50% of the incomes coming from any Copy. That's why

Figure 77 depicts that 50% of the 20$ perceived by the Instantiator is transmitted to the Adaptor, which supposes 10$. Moreover, as the Copy is considered a derivative from the Instance, which is on its turn a derivative from the Adaptation, and the Creator transferred only the 20% of the rights over derivatives (ROD) in License 1, this means that the Creator preserves the 80% of the incomes coming from any Instance. That's why

Figure 77 depicts that 80% of the 10$ perceived by the Adaptor is transmitted to the Creator i.e. 8$.

6.6 Conclusions

Currently, the Linked-Work system is being commercialised by NetPortedItems S.L. company [71] under the name of IPOS-DS (Intellectual Property Operations System – Digital Shadow). Linked-Work is accessible for the general public since April 2008. Everyone can access the web application after an online registration process, accessible from Linked-Work main page [71]. It is currently offered free of charge.

The goal for the next months is to promote its usage amongst different user communities that may be interested in using such a system for spreading their works and creations in an environment of other users that wish to operate within a mutual trust environment in a predictable way.

One of the potential groups where the system may be of interest could be the composer's collective, where different users with a trusted relationship use to collaborate to create, arrange and instantiate audio or audiovisual content.

Another potential goal to take into account could be the adoption of the Linked-Work platform by the collecting societies in different countries. The Linked-Work system could help to spread and ease the management of content generated by the millions of creators, adaptors and instantiators around the world generating content without any collective management or even digital object governance for that matter, as is the case with Creative Commons licenses [40]. Thus, the adoption of IPOS by the collecting societies would provide much added value by offering their constituents and other users that later may become members the benefit of their collective management services.

The advantage of using Linked-Work instead of Creative Commons is that the author is able to establish links with users anywhere to determine exactly the conditions under which their content may be used or accessed. Moreover, by means of Linked-Work revocation functionality and unlike Creative Commons, any author may decide in any moment to stop offering previously determined conditions of use of their content and define a new set of potential rights to be applied from that moment onwards. The revocation functionality does not apply for sure over any license that could have been acquired prior to the revocation. However, authors, by means of temporal or usage limit restrictions can always keep control of the rights they have commercialised prior to the revocation moment, avoiding e.g. an unlimited and unrestricted usage of their content. Linked-Work, by means of its related services, offers other additional functionality with respect to Creative Commons, such as the generation of reports directed to all users corresponding to the object’s ancestry.

Part IV. Conclusion

7 Conclusions and Future Work

7.1 Conclusions

The hypotheses of the research presented in this work have been correctly posed. A generic architecture for the management and protection of multimedia information has been defined and implemented. For this purpose, several standards and solutions have been analysed and taken into account. A relevant standard that has been considered is MPEG-21, together with other *de facto* standards and initiatives explained in the State of the Art.

The first steps towards the definition of the generic DRM architecture have been based on the integration of different parts of the MPEG-21 standard. In this sense, the first contribution has involved the integration of DID an REL parts, identifying how rights expressions can be embedded and extracted from the digital objects to which they apply. The second contribution in this sense integrates the previous DID and REL work together with the DIP part. The DIP part enables the management of multimedia information related to many of the MPEG-21 parts, which goes a step further in the definition of a DRM system.

The results of these contributions consist, on one hand, in a software, included in the reference software part of the standard [26] and a core experiment, which is the mechanism provided by MPEG-21 to investigate if new functionalities need to be added to the standard and the best solutions to be adopted. The results obtained in the DIP core experiment have influenced the edition of the standard, where two new interfaces have been accepted and added to the DIP specification [6].

After the integration tasks in the MPEG-21 standard have been completed, the definition of a MPEG-21-based architecture has been tackled. This architecture is the result of the work performed in the integration of the MPEG-21 parts.

The next step that has been tackled is the generalisation of the initial MPEG-21-based architecture so that it can be generic enough so as to deal with different formats and support the requirements of different standards and initiatives. For this purpose, several standards and solutions have been considered. The architecture has been called Multimedia Information Protection and Management System (MIPAMS). We have presented the integral components is consists of, we have analysed in detail the functionality provided by each of the components and provided different use cases where to prove its feasibility. After this, we have seen how other initiatives can be mapped to MIPAMS, highlighting the differences between them and performing their comparison at different levels. The results demonstrate that MIPAMS architecture is generic enough to cover the functionality present in other architectures.

In the same chapter, we have presented different implementations that have been developed, which are based on MIPAMS or tightly related to it. Those implementations have been developed in different projects such as VISNET II FP6 Network of Excellence [19], AXMEDIS FP6 Integrated Project [21] and GILDDA Spanish Project [20] funded by the Spanish MITyC. In VISNET II section, we have presented how the architecture is being deployed to cover a virtual collaboration scenario. In GILLDA section, we have focused on the usage of MIPAMS in the publishing world. In AXMEDIS section, we have provided a thorough analysis of the DRM architecture, the security measures that have been designed and developed and the testing procedure that has been propose and tackled for the verification and validation of the developed solution.

Finally, we have presented Linked-Work, a specific architecture devised for the registration of original and derived digital content while providing some Digital Rights Management features that enable the creation of trust chains along the multimedia content value chain. Although we have proved that Linked-Work can be mapped into MIPAMS generic architecture, we have seen that it is more focused on the content creation and derivation, and it provides advanced functionality that is not usually present in a DRM architecture. In Linked-Work we have presented new concepts such as: 1) the link between original and derived works; 2) the definition of potential rights and conditions over the Linked-Work objects; 3) the transmission and inheritance models for the potential rights; 4) the inheritance model for event report requests, which enables all the authors in the content value chain to be informed about the actions performed over their Objects or any derivatives; 5) the possibility to keep a percentage of the rights over the derivative objects registered by other authors. Moreover, we have proved how the combination of all these features enables an automatic distribution of the incomes amongst all the authors in the content value chain by means of the definition and analysis of different use cases. Currently, the Linked-Work system is being commercialised by NetPortedItems S.L. company [71] under the name of IPOS-DS (Intellectual Property Operations System – Digital Shadow) and is being offered free of charge for evaluation [71].

As a result of the usage of standards in the developed architectures, some issues regarding the distributed generation of MPEG-21 Event Reporting have arisen. We have presented the proposal we sent MPEG-21 Event Reporting in order to solve the problem that arises when reporting information coming from different sources, regarding which information has been added by which of the parties. This contribution has been considered, accepted and included in the resulting corrigendum [49], having a direct impact on the final text of the standard.

7.2 Publications

The research presented in this work has been validated against the research community in the context of relevant conferences and standardisation initiatives. This section presents the collection of publications related to the research that has been carried out and the impact of the work in the MPEG-21 standard. All the publications compiled in this section have been put into context in the corresponding chapter.

7.2.1 Refereed Publications

Enhancing rights management systems through the development of trusted value networks. Torres, V., Delgado, J., Maroñas, X., Llorente, S., Gauvin, M. ACM International Conference on Multimedia 2008. October 27 – November 1. Vancouver (Canada). Acceptance pending.

This paper describes the work presented in the Linked-Work section, focusing on its real application and commercialisation under the name of IPOS-DS. ACM-MM is one of the top-ranked conferences in the Computer Science area. It is included in the ranking published by the Research and Education Association of Australasia (CORE) with the maximum score.

Event Reporting scenarios and implementations in multimedia content distribution and consumption. Torres, V., Rodríguez, E., Delgado, J. Automated Production of Cross Media Content for Multi-Channel Distribution, 2008. AXMEDIS' 08. Fourth International Conference on. November 17-19. Florence (Italy). Acceptance pending.

This paper presents part of the work developed in the Event Reporting area, focusing on the different implementations of Event Reporting done in different projects such as AXMEDIS, VISNET II and Linked-Work. Regarding Linked-Work implementation it presents some new concepts such as the Event Report Request transmission and inheritance. The AXMEDIS conference is focused on the research, developments and applications in the cross media domain, exploring new and innovative technologies to meet the challenges of the sector. It has brought together the experiences and communities coming from the WEDELMUSIC conference series, the MUSICNETWORK and other co-located workshops.

Reporting Events in a multimedia content distribution and consumption system. Torres, V., Rodríguez, E., Delgado, J. The 14th International Conference on Distributed Multimedia Systems. DMS 2008. September 4 - September 6. Boston (USA). Publication pending.

This paper describes the work presented in the Event Reporting area, focusing on the distributed generation of event reports and use cases. The DMS conference is an international conference series, which covers a wide spectrum of paper presentations, technical discussions and demonstrations in the fields of distributed multimedia computing.

Open DRM and the Future of Media. Torres, V., Serrao, C., Delgado, J., Dias, M. IEEE Multimedia. ISSN: 1070-986X. To be published in the 2008 April-June issue.

This paper gives an overview of the current state of DRM systems and provides a comparison of three systems, selected for being open source, open specification and open interfaces. I also provide some hints on the interoperability of DRM systems. IEEE MultiMedia is an International Journal that covers several fields such as image processing, video processing, audio analysis, text retrieval and understanding, data mining and analysis and data fusion. It is included in the ISI Journal Citation Reports (JCR).

Trusting software tools in a secure DRM architecture. Torres, V., Delgado, J., Llorente, S. Automated Production of Cross Media Content for Multi-Channel Distribution, 2007. AXMEDIS'07. Third International Conference on. 28-30 November 2007. Barcelona (Spain). IEEE Computer Society, pp. 55-61, 2007. ISBN: 978-0-7695-3030-7.

This paper presents a part of the MIPAMS security mechanisms developed in the AXMEDIS DRM architecture, focusing on the recertification of tools. The AXMEDIS conference is focused on the research, developments and applications in the cross media domain, exploring new and innovative technologies to meet the challenges of the sector. It has brought together the experiences and communities coming from the

WEDELMUSIC conference series, the MUSICNETWORK and other co-located workshops.

Interoperability Mechanisms for registration and authentication on different Open DRM platforms. Serrao, C., Torres, V., Delgado, J., Dias, M. IJCSNS International Journal of Computer Science and Network Security, Vol. 6 No.12, pp. 291-303, 2006. ISSN: 1738-7906.

This paper focuses on the comparison of two DRM architectures, i.e. MIPAMS and OpenSDRM, from a security perspective. The paper also proposes a generic solution in order to make different DRM architectures interoperable at the security level. The International Journal of Computer Science and Information Security is a monthly journal that publishes articles which contribute new theoretical results in all areas of Computer Science, Communication Network and Information Security.

An implementation of a trusted and secure DRM architecture. Torres V. Delgado, J., Llorente, S. On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops. Lecture Notes in Computer Science (LNCS), Vol. 4277, pp. 312-321, 2006. Springer Berlin Heidelberg New York. ISBN-10: 3-540-48273-3. ISBN-13: 978-3-540-48273-4. ISSN: 0302-9743.

This paper presents a part of the MIPAMS security mechanisms developed in the AXMEDIS DRM architecture, focusing on user authentication, tool registration, tool certification and tool verification. It describes the mechanisms needed to have a trusted system both in server and client parts. This paper was submitted to the First International Workshop on Information Security (IS'06), in conjunction with OnTheMove Federated Conferences (OTM'06), the fifth edition of the OTM Federated Conferences. The IS’06 workshop had a highly selective acceptance ratio (under 25%).

Use of standards for implementing a Multimedia Information Protection and Management System. Torres, V., Rodríguez, E, Llorente, S., and Delgado, J. Automated Production of Cross Media Content for Multi-channel Distribution, 2005. AXMEDIS’05. First International Conference on. 30 November – 2 December. Florence (Italy). IEEE Computer Society, pp. 145-153, 2005. ISBN: 0-7695-2348-X.

This paper describes the MIPAMS architecture, provides a mapping to the MPEG-21 standard and OMA DRM initiative and includes a section about how the implementation of standards and the AXMEDIS implementation. The AXMEDIS conference is focused on the research, developments and applications in the cross media domain, exploring new and innovative technologies to meet the challenges of the sector. It has brought together the experiences and communities coming from the WEDELMUSIC conference series, the MUSICNETWORK and other co-located workshops.

Rights and Trust in Multimedia Information Management. Delgado, J., Torres, V., Llorente, S. and Rodríguez, E. 9th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security. CMS 2005. September 19-21. Viena (Austria). Lecture Notes in Computer Science, Vol. 3677 (2005), pp. 55-64. ISBN: 3-540-28791-4. ISSN: 0302-9743.

This paper describes the MIPAMS architecture and includes a section detailing a content consumption use case, where the MIPAMS functionality is analysed in detail. A specific section is included where the development of tools and components related to the MIPAMS functionalities is described. The CMS conference is a joint working conference of IFIP TC6 and TC11. The conference had a highly selective acceptance ratio (19,60%), where only 28 papers of 143 were accepted.

Trust and rights in multimedia content management systems. Torres, V., Rodríguez, E, Llorente, S., and Delgado, J. Proceedings of the IASTED International Conference Web Technologies, Applications, and Services. WTAS 2005. July 4-6. Calgary (Canada). ACTA Press, Anaheim Calgary Zurich, pp. 89-94, 2005. ISBN: 0-88986-483-7.

This paper introduces the initial MPEG-21-based architecture and explains how it has been extended and generalised so as to define the MIPAMS architecture. It includes a section regarding the relationship of the architecture components to standards, a description of how the components provide trust to the whole system and a section presenting the implementation plan. Web Technologies, Applications, and Services 2005 (WTAS 2005) is the first of a series of conferences organised by the International Association of Science and Technology for Development (IASTED).

Architecture and Protocols for the Protection and Management of Multimedia Information. Torres, V., Rodríguez, E, Llorente, S., and Delgado, J. Second International Workshop on Multimedia Interactive Protocols and Systems. MIPS 2004. November 16-19. Grenoble (France). Lecture Notes in Computer Science (LNCS), Vol. 3311/2004, pp. 252–263, 2004. Springer Berlin Heidelberg New York. ISBN-10: 3-540-23928-6. ISSN: 0302-9743.

This paper defines the initial MPEG-21-based architecture and describes a use case based on the execution of a MPEG-21 Digital Item Method (DIM). The International Workshop on Multimedia Interactive Protocols and Systems (MIPS 2004) is the result of the combination of two conferences: Interactive Distributed Multimedia Systems (IDMS) and Protocols for Multimedia Systems (PROMS). It continues being held under the name of CoNEXT. The conference had a highly selective acceptance ratio (33%), where only 25 papers of 74 were accepted, including the short papers.

7.2.2 Standardisation Publications

This section presents the contributions to standardisation bodies, especially to the MPEG-21 standard. The contributions to MPEG-21 have been done in the form of input documents to the Multimedia Description Schemes (MDS) and Requirements groups. They have been presented in the corresponding meetings and after being accepted, they are currently included in the MPEG-21 standard.

Some issues on the generation and modification of Event Reports in the MPEG-21 Event Reporting. Rodríguez, E., Delgado, J., Torres, V. ISO/IEC JTC1/SC29/WG11 MPEG2007/M14508. April 2007.

MPEG-21 DIP Core Experiments: A contribution to the implementation of DIBOs for REL. Torres, V., Delgado, J., Rodríguez, E. ISO/IEC JTC 1/SC 29/WG 11/M10873. July 2004.

Status of DMAG Reference Software for MPEG 21 REL, RDD and DID. Delgado, J. Rodríguez, E., Llorente, S., García, R., Torres, V. ISO/IEC JTC 1/SC 29/WG 11/M10577. March 2004.

Contribution to DID/REL/RDD reference software: DMAG REL License Interpretation within DID using RDD term genealogy. Torres, V., Delgado, J., Rodríguez, E. ISO/IEC JTC 1/SC 29/WG 11/M10575. March 2004.

DMAG REL License Interpretation within DID. Torres, V., Delgado, J., Rodríguez, E. ISO/IEC JTC 1/SC 29/WG 11/M10421. December 2003.

7.3 Future Work

A future research line would be that of following the market evolution to determine how it influences the success or failure of existing DRM systems and architectures. Currently there are many systems in the market and there is no chance of success for all of them. An example of failure is that of Melodeo's [84] PachyDRM, which started providing a DRM solution for the mobile world and whose website has been recently closed. Moreover, there are some standards, such as MPEG-21, whose scope is so wide that it is difficult to implement all the features they specify.

In this context, MPEG's Multimedia Application Formats (MAF) are a good candidate to take into account, as they provide the framework for integration of elements from several MPEG standards into a single specification that is suitable for specific, but widely usable applications. Typically, MAFs specify how to combine metadata with timed media information for a presentation in a well-defined format that facilitates interchange, management, editing, and presentation of the media. Typically, MAF specifications include: the ISO File Format family for storage, a simple MPEG-7 tool set for Metadata, one or more coding Profiles for representing the Media and tools for encoding metadata in either binary or XML form. Moreover, MAFs may specify use of: MPEG-21 Digital Item Declaration Language for representing the Structure of the Media and the Metadata, other MPEG-21 tools, non-MPEG coding tools (e.g., JPEG) for representation of "non-MPEG" media and elements from non-MPEG standards that are required to achieve full interoperability. MAF Specifications can contain elements from all existing MPEG Standards and make use of the profiles they define, which is an easy way to cope with a subset of the standard.

Anyway, DRM convergence is something that will happen in the future, when the different DRM systems that survive will need to interoperate to satisfy the user's needs and requirements. Therefore, the work presented in this document can be used as the starting point for identifying the common points between the different alternatives and design interoperability mechanisms amongst them.

Another future research line would be that of Linked-Work system. Several aspects need to be considered in this line.

First, it would be very useful to analyse how Linked-Work can be integrated with existing collecting societies systems in order to provide them an alternative way of managing the intellectual property of people's creations, while extending the potential number of customers they manage by means of the digital technologies.

Second, we need to analyse how the technology and concepts implemented in Linked-Work can be applied to other areas or environments. The registration and certification functionality provided in Linked-Work together with the linkage of the original and derived content could be e.g. extrapolated for controlling the pedigree and ownership of any kind of products that are involved in a production chain.

Third, another desirable feature in Linked-Work would be the possibility of integrating the available functionality as a part of other system's processes, which could use Linked-Work services integrated in an automated process. In general, the Linked-Work functionality is easily automatable, as it is implemented using the web services approach. Thus, web services can be easily orchestrated, but there are some parts that would need to be adapted to ease the process.

Finally, other aspects could be considered for the improvement of current system, such as the development of user communities, where the social relationship between Linked-Work members needs to be strengthened, the consideration of usability issues and the addition new functionality as e.g. the possibility of registering surrogate objects for being able to create new derived objects from parent objects that are not registered in the system. This latter feature should be developed with the help and collaboration of collecting societies for those objects that are out of the public domain.

List of Acronyms

AXCS

AXMEDIS Certifier and Supervisor.

AXCV

AXMEDIS Certification and Verification.

AXMEDIS

Automating Production of Cross Media Content for Multichannel Distribution.

DI

Digital Item. Structured digital object, including a standard representation, identification and meta-data within the MPEG-21 framework. This entity is the fundamental unit of distribution and transaction within the multimedia framework as a whole.

DIA

Digital Item Adaptation as specified by ISO/IEC 21000-7.

DIBO

Digital Item Base Operation. Base operation providing access to functionality implemented by a Peer and used in authoring a Digital Item Method specified by ISO/IEC 21000-10.

DID

Digital Item Declaration. Declaration of the resources, metadata and their interrelationships of a Digital Item specified by ISO/IEC 21000-2.

DIDL

Digital Item Declaration Language. XML-based language including validation rules specified by ISO/IEC 21000-2 for the standard representation in XML of a Digital Item Declaration.

DIDL document

A document using the Digital Item Declaration Language to declare a Digital Item in a standard representation in XML specified by ISO/IEC 21000-2.

DIDL element

XML element of the Digital Item Declaration Language specified by ISO/IEC 21000-2.

DID Model

Set of abstract terms and concepts specified by ISO/IEC 21000-2 forming a model for declaring Digital Items.

DIXO

Digital Item eXtension Operation. Operation allowing extended functionality to be invoked from a Digital Item Method specified by ISO/IEC 21000-10.

DCI

The Digital Media Project (DMP) Content Information.

DCF

Open Mobile Alliance (OMA) DRM Content Format.

DII

Digital Item Identification. Digital Item Identification as specified by ISO/IEC 21000-3.

DIM

Digital Item Method. Tool for expressing the suggested interaction of a User with a Digital Item at the level of the Digital Item Declaration specified by ISO/IEC 21000-10.

DIML

Digital Item Method Language. Language providing the syntax and structure for authoring a Digital Item Method utilizing the Digital Item Base Operations specified by ISO/IEC 21000-10.

DMAG

Distributed Multimedia Applications Group.

DMP

The Digital Media Project.

DOM

Document Object Model Level 1, 2 and 3, W3C Recommendations.

DRM

Digital Right Management.

End User

User taking the role of consumer, i.e. being at the end of a value or delivery chain. For example, a human consumer, an agent operating on behalf or a human consumer, etc.

EBNF

Extended Backus-Naur Form.

ER

ISO/IEC 21000:15:2006 Event Report.

ERR

ISO/IEC 21000:15:2006 Event Report Request.

GUI

Graphical User Interface.

IPMP

Intellectual Property Management and Protection as specified by ISO/IEC 21000-4.

JPEG

Joint Photographic Experts Group.

MIME

Multipurpose Internet Mail Extensions (see IETF RFC 2045).

MIPAMS

Multimedia Information Protection and Management System.

MP3

MPEG-1/2 layer 3 (audio coding).

MPEG

Moving Picture Experts Group.

MPEG-21

ISO/IEC 21000 (all parts).

Namespace

XML namespace, W3C Recommendation.

ODRL

Open Digital Rights Language.

OMA

Open Mobile Alliance.

OpenSDRM

Open and Secure Digital Rights Management platform.

Peer

Device or application that compliantly processes a Digital Item.

PKI

Public-key Infrastructure.

PMS

AXMEDIS Protection Management Support.

RDD

Rights Data Dictionary as specified by ISO/IEC 21000-6.

RRD

Rights Represent Data specified by the Digital Media Project (DMP).

REL

Rights Expression Language.

SAML

Security Assertion Markup Language, OASIS standard.

SoA

Service-oriented Architecture.

SOAP

Simple Object Access Protocol, W3C Recommendation.

SSL/TLS

Secure Socket Layer / Transport Layer Security (see IETF RFC 2246).

UDDI

Universal Description, Discovery, and Integration, Organization for the Advancement of Structured Information Standards (OASIS) standard.

UUID

Universally Unique Identifier, Open Software Foundation (OSF) standard.

URI

Uniform Resource Identifier (see IETF RFC 3986).

URL

Uniform Resource Locator (see IETF RFC 1738).

URN

Uniform Resource Name (see IETF RFC 2141).

User

Entity that interacts in the MPEG-21 environment or makes use of Digital Items.

WSDL

Web Services Description Language, W3C Note.

W3C

World Wide Web Consortium.

XML

Extensible Markup Language, W3C Recommendation.

XMLDSIG

XML-Signature Syntax and Processing, W3C Recommendation.

XMLENC

XML-Encryption Syntax and Processing, W3C Recommendation.

XMLSCHEMA

XML Schema Part 1: Structures and Part 2: Datatypes, W3C Recommendation.

XPath

XML Path Language, W3C Recommendation.

XQuery

XML Query Language, W3C Working Draft.

References

[1] Distributed Multimedia Applications Group (DMAG). http://dmag.upf.edu/, http://recerca.ac.upc.edu/DMAG/

[2] MPEG-21 Standard. http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm.

[3] ISO/IEC 21000-2:2005, Information technology – Multimedia framework (MPEG-21) – Part 2: Digital Item Declaration

[4] ISO/IEC 21000-5:2004, Information technology – Multimedia framework (MPEG-21) – Part 5: Rights Expression Language.

[5] ISO/IEC 21000-6:2004, Information technology – Multimedia framework (MPEG-21) – Part 6: Rights Data Dictionary.

[6] ISO/IEC 21000-10:2006, Information technology – Multimedia framework (MPEG-21) – Part 10: Digital Item Processing.

[7] Open Mobile Alliance, DRM Specification (OMA-TS-DRM_DRM-V2_0_1-20080226-A), February 2008, http://www.openmobilealliance.org/Technical/release_program/drm_v2_0.aspx.

[8] DReaM-CAS Client Specification. Version 0.9c. Technical Specification. March 2006. DReaM-MMI Specification. Sun Microsystems Laboratories – Sun Labs. Version 0.8. March 2006. http://www.openmediacommons.org/. https://dream.dev.java.net/.

[9] Marlin Architecture Overview. Marlin Developer Community. http://www.marlin-community.com/.

[10] The Role of Octopus in Marlin. Marlin Developer Community. 2006.

[11] OpenSDRM. http://sourceforge.net/projects/opensdrm/.

[12] Windows Media DRM. http://www.microsoft.com/windows/windowsmedia/forpros/drm/default.mspx, http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx.

[13] Windows Media DRM. http://www.microsoft.com/windows/windowsmedia/drm/9series/providers.aspx.

[14] iTunes. http://www.apple.com/itunes/.

[15] Coral Consortium Core Architecture Overview. CCAWC Editors. June 2006. CCAWC Core Architecture. CCAWC Editors. June 2006. Ecosystem-A Specification. CCAWC Editors. June 2006. http://www.coral-interop.org/.

[16] Coral Consortium. An Overview. February 2006. Coral Consortium Corporation.

[17] CCAWC Core Architecture. June 2006. Coral Consortium Architecture Specification 3.0. Coral Consortium Architecture Working Committee. Coral Consortium Corporation.

[18] The Digital Media Project. 2008. http://www.dmpf.org/.

[19] NETworked audioVISual media technologies (VISNET II), FP6-2005-IST-41. http://www.visnet-noe.org/.

[20] Gestión Integral para el Libro Digital: Derechos de Autor, contenidos y negocio (GILDDA). Spanish research project. Ministerio de Industria, Turismo y Comercio (Programa Tractor).

[21] Automatic Production of Cross Media Content for Multichannel Distribution (AXMEDIS), IST 2004 511299. http://www.axmedis.org/.

[22] ISO/IEC 21000-1:2004, Information technology – Multimedia framework (MPEG-21) – Part 1: Vision, Technologies and Strategy.

[23] ISO/IEC 21000-3:2003, Information technology – Multimedia framework (MPEG-21) – Part 3: Digital Item Identification.

[24] ISO/IEC 21000-4:2006, Information technology – Multimedia framework (MPEG-21) – Part 4: Intellectual Property Management and Protection Components.

[25] ISO/IEC 21000-7:2007, Information technology – Multimedia framework (MPEG-21) – Part 7: Digital Item Adaptation.

[26] ISO/IEC 21000-8:2008, Information technology – Multimedia framework (MPEG-21) – Part 8: Reference Software.

[27] ISO/IEC 21000-9:2005, Information technology – Multimedia framework (MPEG-21) – Part 9: File Format.

[28] ISO/IEC 21000-11:2004, Information technology – Multimedia framework (MPEG-21) – Part 11: Evaluation Tools for Persistent Association Technologies.

[29] ISO/IEC 21000-12:2005, Information technology – Multimedia framework (MPEG-21) – Part 12: Test Bed for MPEG-21 Resource Delivery.

[30] ISO/IEC 21000-14:2007, Information technology – Multimedia framework (MPEG-21) – Part 14: Conformance Testing.

[31] ISO/IEC 21000-15:2006, Information technology – Multimedia framework (MPEG-21) – Part 15: Event Reporting.

[32] ISO/IEC 21000-16:2005, Information technology – Multimedia framework (MPEG-21) – Part 16: Binary Format.

[33] ISO/IEC 21000-17:2006, Information technology – Multimedia framework (MPEG-21) – Part 17: Fragment Identification of MPEG Resources.

[34] ISO/IEC 21000-18:2007, Information technology – Multimedia framework (MPEG-21) – Part 18: Digital Item Streaming.

[35] eXtensible rights Markup Language (XrML). http://www.xrml.org/.

[36] Wang, X., Delgado, J., Barlas C. “ISO/IEC 21000-5/FDAM 1 Rights Expression Language: the MAM profile”. ISO/IEC JTC 1/SC 29/WG 11/N8342. Klagenfurt, Austria, July 2006

[37] Chiariglione, F., Kim, T., Wang, X. “ISO/IEC 21000-5/FDAM 2 Rights Expression Language: the DAC profile”. ISO/IEC JTC 1/SC 29/WG 11/N8812. Marrakech, MA, February 2007.

[38] TV-Anytime Forum. http://www.tv-anytime.org/.

[39] Kim, T., Delgado, J., Schreiner, F., Barlas C., Wang, X. “ISO/IEC 21000-5:2004/FPDAM 3: ORC (Open Release Content) Profile” ISO/IEC JTC 1/SC 29/WG 11/N9108. April 2007, San Jose, USA.

[40] Creative Commons. http://creativecommons.org/.

[41] ISO/IEC, ISO/IEC SoCD 21000-10 – Digital Item Processing.

[42] ECMAScript Compact Profile, ECMA-327, 3rd Ed., ECMA General Assembly, June 2001.

[43] Open Mobile Alliance, DRM Rights Expression Language (OMA-TS-DRM_REL-V2_0_1-20080226-A), February 2008. http://www.openmobilealliance.com/Technical/ release_ program/drm_v2_0.aspx.

[44] Open Digital Rights Language (ODRL). http://odrl.net/.

[45] XML Encryption. http://www.w3.org/Encryption/2001/.

[46] XML-Signature Syntax and Processing. http://www.w3.org/TR/xmldsig-core/.

[47] The Digital Media Project. Source GA15. Date 2007/07/20. Approved Document No. 3 – Technical Specification: Interoperable DRM Platform, Version 3.0. No.1003/GA15.

[48] Chillout. http://chillout.dmpf.org/.

[49] Timmerer, C., Delgado, J. ISO/IEC 21000-15:2006/DCOR 1 MPEG-21 Event Reporting. ISO/IEC JTC1/SC29/WG11 MPEG2007/N9118. April 2007.

[50] “DMAG REL License Interpretation within DID”. Torres, V., Delgado, J., Rodríguez, E. ISO/IEC JTC 1/SC 29/WG 11/M10421. December 2003. Hawaii, USA.

[51] “Contribution to DID/REL/RDD reference software: DMAG REL License Interpretation within DID using RDD term genealogy”. Torres, V., Delgado, J., Rodríguez, E. ISO/IEC JTC 1/SC 29/WG 11/M10575. Munich, Germany. March 2004.

[52] “MPEG-21 REL/RDD Software Implementation Plan v.4”. ISO/IEC JTC 1/SC 29/WG 11/N5931. October 2003. Brisbane, Australia.

[53] “DMAG REL License Interpreter v.1.1”. ISO/IEC JTC 1/SC 29/WG 11/M10038. October 2003. Brisbane, Australia.

[54] Getting Kawa. http://www.gnu.org/software/kawa/Getting-Kawa/.

[55] XML Query. http://www.w3.org/XML/Query/.

[56] “DMAG REL license interpreter using RDD term genealogy – implementation with web services”. ISO/IEC JTC 1/SC 29/WG 11/M10574. March 2004. Munich, Germany.

[57] "MPEG Ontologies API". ISO/IECJTC1/SC29/WG11/M10702. March 2004, Munich.

[58] “Contribution to DID 2nd Edition reference software”. ISO/IEC JTC 1/SC 29/WG 11/M10609. Munich, Germany. March 2004.

[59] “MPEG-21 DIP Core Experiments: A contribution to the implementation of DIBOs for REL”. Torres, V., Delgado, J., Rodríguez, E. ISO/IEC JTC 1/SC 29/WG 11/M10873. Redmond, USA. July 2004.

[60] “Workplan for Core Experiment on DIBOs for REL”. ISO/IEC JTC 1/SC 29/WG 11/N6418. Munich, Germany. March 2004.

[61] “MPEG-21 REL/RDD Software Implementation Plan v.5”. ISO/IEC JTC 1/SC 29/WG 11/N6165. December 2003. Hawaii, USA.

[62] “DMAG Schema Checker”, ISO/IEC JTC 1/SC 29/WG 11/M10039. Brisbane, Australia. October 2003.

[63] “DMAG REL Validation Rules Checker”, ISO/IEC JTC 1/SC 29/WG 11/M10041. Brisbane, Australia. October 2003.

[64] Rodríguez, E., Delgado, J., Torres, V. Some issues on the generation and modification of Event Reports in the MPEG-21 Event Reporting. ISO/IEC JTC1/SC29/WG11 MPEG2007/M14508. April 2007.

[65] Serrão C., Serra A., Dias M., Delgado J., "Protection of MP3 Music Files Using Digital Rights Management and Symmetric Ciphering", Proceedings of the 2nd International Conference on Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS2006), Leeds, UK, 13-15 December, 2006.

[66] A Universally Unique IDentifier (UUID) URN Namespace. http://tools.ietf.org/html/4122/.

[67] Internet Assigned Numbers Entity (IANA) Private Enterprise Number. http://www.iana.org/cgi-bin/enterprise.pl.

[68] PKCS #12 v1.0: Personal Information Exchange Syntax Standard. RSA Laboratories, June 24, 1999. http://www.rsasecurity.com/.

[69] The Digital Media Project. 2007. Approved Document No 2 – Technical Reference: Architecture, Version 3.0. No.1002/GA15.

[70] Gauvin, M., Delgado, J., Rodríguez, V. Proposed RRD Text for Approved Document No 3 – Technical Specification: Interoperable DRM Platform, ver. 2.1. No. 0953/AHG41

[71] NetPortedItems. http://www.digitalmediavalues.com/.

[72] Dublin Core. Dublin Core Metadata Initiative (DCMI). http://dublincore.org/.

[73] Torres, V., Rodríguez, E., Delgado, J. Event Reporting scenarios and implementations in multimedia content distribution and consumption. Automated Production of Cross Media Content for Multi-Channel Distribution, 2008. AXMEDIS' 08. Fourth International Conference on. November 17-19. Florence (Italy). Acceptance pending.

[74] Torres, V., Rodríguez, E., Delgado, J. Reporting Events in a multimedia content distribution and consumption system. The 14th International Conference on Distributed Multimedia Systems. DMS 2008. September 4 - September 6. Boston (USA). Acceptance pending.

[75] Torres, V., Rodríguez, E, Llorente, S., and Delgado, J. Architecture and Protocols for the Protection and Management of Multimedia Information. Second International Workshop on Multimedia Interactive Protocols and Systems. MIPS 2004. November 16-19. Grenoble (France). Lecture Notes in Computer Science (LNCS), Vol. 3311/2004, pp. 252–263, 2004. Springer Berlin Heidelberg New York. ISBN-10: 3-540-23928-6. ISSN: 0302-9743 .

[76] Torres, V., Serrao, C., Delgado, J., Dias, M. Open DRM and the Future of Media. IEEE Multimedia. ISSN: 1070-986X. To be published in the 2008 April-June issue.

[77] Torres, V., Delgado, J., Llorente, S. Trusting software tools in a secure DRM architecture. Automated Production of Cross Media Content for Multi-Channel Distribution, 2007. AXMEDIS'07. Third International Conference on. 28-30 November 2007. Barcelona (Spain). IEEE Computer Society, pp. 55-61, 2007. ISBN: 978-0-7695-3030-7.

[78] Serrao, C., Torres, V., Delgado, J., Dias, M.. Interoperability Mechanisms for registration and authentication on different Open DRM platforms. International Journal of Computer Science and Network Security (IJCSNS), Vol. 6 No.12, pp. 291-303, 2006. ISSN: 1738-7906.

[79] Torres V. Delgado, J., Llorente, S. An implementation of a trusted and secure DRM architecture. On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops. Lecture Notes in Computer Science (LNCS), Vol. 4277, pp. 312-321, 2006. Springer Berlin Heidelberg New York. ISBN-10: 3-540-48273-3. ISBN-13: 978-3-540-48273-4. ISSN: 0302-9743.

[80] Torres, V., Rodríguez, E, Llorente, S., and Delgado, J. Use of standards for implementing a Multimedia Information Protection and Management System. Automated Production of Cross Media Content for Multi-channel Distribution, 2005. AXMEDIS'05. First International Conference on. 30 November – 2 December. Florence (Italy). IEEE Computer Society, pp.145-153, 2005. ISBN: 0-7695-2348-X.

[81] Delgado, J., Torres, V., Llorente, S. and Rodríguez, E. Rights and Trust in Multimedia Information Management. 9th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security. CMS 2005. September 19-21. Viena (Austria). Lecture Notes in Computer Science, Vol. 3677 (2005), pp. 55-64. ISBN: 3-540-28791-4. ISSN: 0302-9743.

[82] Torres, V., Rodríguez, E, Llorente, S., and Delgado, J. Trust and rights in multimedia content management systems. Proceedings of the IASTED International Conference Web Technologies, Applications, and Services. WTAS 2005. July 4-6. Calgary (Canada). ACTA Press, Anaheim Calgary Zurich, pp. 89-94, 2005. ISBN: 0-88986-483-7.

[83] Torres, V., Delgado, J., Maroñas, X., Llorente, S., Gauvin, M. Enhancing rights management systems through the development of trusted value networks. ACM International Conference on Multimedia 2008. October 27 – November 1. Vancouver (Canada). Acceptance pending.

[84] http://www.nutsie.com/melodeo