

Coral Consortium Application Note

Contribution Number:
Working Committee: Coral Consortium Architecture Working Committee
Title: Providing Interoperability with Windows Media® DRM
Version: 1.0.1
Editor: CCAWC
Date: 28 July 2006

THE CORAL CONSORTIUM ON BEHALF OF ITSELF AND ITS MEMBERS MAKES NO REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED, CONCERNING THE COMPLETENESS, ACCURACY, OR APPLICABILITY OF ANY INFORMATION CONTAINED IN THIS APPLICATION NOTE. THE CORAL CONSORTIUM, FOR ITSELF AND THE MEMBERS, DISCLAIM ALL LIABILITY OF ANY KIND WHATSOEVER, EXPRESS OR IMPLIED, ARISING OR RESULTING FROM THE RELIANCE OR USE BY ANY PARTY OF THIS APPLICATION NOTE OR ANY INFORMATION CONTAINED HEREIN.

THE CORAL CONSORTIUM ON BEHALF OF ITSELF AND ITS MEMBERS MAKES NO REPRESENTATIONS CONCERNING THE APPLICABILITY OF ANY PATENT, COPYRIGHT OR OTHER PROPRIETARY RIGHT OF A THIRD PARTY TO THIS APPLICATION NOTE OR ITS USE, AND THE RECEIPT OR ANY USE OF THIS APPLICATION NOTE OR ITS CONTENTS DOES NOT IN ANY WAY CREATE BY IMPLICATION, ESTOPPEL OR OTHERWISE, ANY LICENSE OR RIGHT TO OR UNDER ANY CORAL CONSORTIUM MEMBER COMPANY'S PATENT, COPYRIGHT, TRADEMARK OR TRADE SECRET RIGHTS WHICH ARE OR MAY BE ASSOCIATED WITH THE IDEAS, TECHNIQUES, CONCEPTS OR EXPRESSIONS CONTAINED HEREIN.

Copyright © 2006 by the Coral Consortium Corporation. All rights reserved.

Coral Consortium Corporation

39355 California Street, Suite 307
Fremont, CA 94538

Tel: +1 510 744 4022
Fax: +1 510 608 5917

www.coral-interop.org

Providing Interoperability with Windows Media® DRM

28 July 2006

Version 1.0.1

THE CORAL CONSORTIUM ON BEHALF OF ITSELF AND ITS MEMBERS MAKES NO REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED, CONCERNING THE COMPLETENESS, ACCURACY, OR APPLICABILITY OF ANY INFORMATION CONTAINED IN THIS APPLICATION NOTE. THE CORAL CONSORTIUM, FOR ITSELF AND THE MEMBERS, DISCLAIM ALL LIABILITY OF ANY KIND WHATSOEVER, EXPRESS OR IMPLIED, ARISING OR RESULTING FROM THE RELIANCE OR USE BY ANY PARTY OF THIS APPLICATION NOTE OR ANY INFORMATION CONTAINED HEREIN.

THE CORAL CONSORTIUM ON BEHALF OF ITSELF AND ITS MEMBERS MAKES NO REPRESENTATIONS CONCERNING THE APPLICABILITY OF ANY PATENT, COPYRIGHT OR OTHER PROPRIETARY RIGHT OF A THIRD PARTY TO THIS APPLICATION NOTE OR ITS USE, AND THE RECEIPT OR ANY USE OF THIS APPLICATION NOTE OR ITS CONTENTS DOES NOT IN ANY WAY CREATE BY IMPLICATION, ESTOPPEL OR OTHERWISE, ANY LICENSE OR RIGHT TO OR UNDER ANY CORAL CONSORTIUM MEMBER COMPANY'S PATENT, COPYRIGHT, TRADEMARK OR TRADE SECRET RIGHTS WHICH ARE OR MAY BE ASSOCIATED WITH THE IDEAS, TECHNIQUES, CONCEPTS OR EXPRESSIONS CONTAINED HEREIN.

Copyright © 2006 by the Coral Consortium Corporation. All rights reserved.

Coral Consortium Corporation

39355 California Street, Suite 307
Fremont, CA 94538

Tel: +1 510 744 4022

Fax: +1 510 608 5917

www.coral-interop.org

Table of Contents

Executive Summary	1
1 Introduction	2
2 The Coral Interoperability Model	3
2.1 Management of Rights	4
2.1.1 Rights Tokens	5
2.1.2 Mapping and Standardization	6
2.1.2.1 Mapping Between Namespaces	6
2.1.2.2 Standardized Namespaces	8
2.1.3 Deriving Licenses	8
2.2 Rights Mediation	9
2.3 Nodes, Roles, and Interfaces	10
3 Windows Media DRM	11
3.1 Windows Media DRM SDKs	12
3.2 Deploying Windows Media DRM	13
3.2.1 Division of Labor	13
3.2.2 Client Side Integration	15
3.2.3 Service Side Integration	16
3.2.4 A Windows Media Walkthrough	17
4 Integrating Coral and Windows Media	19
4.1 Fulfilling Rights Acquired Elsewhere	20
4.1.1 Walkthrough	21
4.1.2 What Has Changed?	23

PROVIDING INTEROPERABILITY WITH WINDOWS MEDIA[®] DRM

4.2	Providing Rights to Others	23
5	Conclusions	25
6	References	26

Executive Summary

The Coral Consortium approach to DRM interoperability is based on the integration of a thin layer of standardized technologies with existing DRM systems. This application note describes the integration of the Coral interoperability framework with Windows Media® DRM (WM-DRM), one of the most widely-used systems for protecting electronically distributed content. Integration with Coral gives systems based on WM-DRM the ability to interoperate with non Windows-based devices and services, allowing adopters to provide transparency and predictability of content usage across DRM systems while relying on standard technology provided by Microsoft and other DRM providers.

Like other DRM technologies available through software development kits (SDKs), WM-DRM leaves certain higher-level technology decisions to adopters. In most cases, DRM SDKs handle the *hows* of a given deployment, but not the *whys*. For example, DRM SDKs do not normally specify the circumstances that cause a DRM client to request a license, nor do they fully define the conditions under which such licenses are granted by a license server. These issues must be decided by deployers of the technology and implemented using the underlying technology provided by the SDKs.

SDK-based DRM systems can be integrated with the Coral interoperability framework in a straightforward way, with little or no modification to the SDKs. Many of the decisions that would normally be defined by adopters and systems integrators in the context of a single DRM technology are augmented by policies from the interoperability framework. In a typical deployment, a WM-DRM license server may grant a license based on a client request only if the request contains information that correlates that request with a previous commercial transaction in which the rights to acquire a license were purchased. In an interoperable context, the commercial transaction that authorizes access to DRM licenses might be replaced in some cases by an interoperability transaction in which rights obtained from an external system were registered with the Coral-enabled license server.

The Coral framework provides a standardized infrastructure for communicating the rights that allow a consumer to access licenses and content in different DRM formats. Some of these components must be integrated directly with native DRM components. For example, Coral defines a *Rights Instantiator* function that interacts with both the Coral framework and with native DRM systems, deriving DRM licenses from standardized Coral data structures. This application note shows that these Coral components can be integrated with WM-DRM in a way that respects the design of the technology and its intended uses, without requiring modifications to the technology.

The ability to integrate naturally with systems based on Windows Media® DRM demonstrates the strength of the Coral approach to interoperability, which acknowledges the

diversity of existing DRM systems and provides a minimal set of standardized integration points that work with these systems and allow them to work with one another.

1 Introduction

Lack of interoperability among DRM systems is one of the major factors inhibiting the development of a rich, competitive electronic media distribution market. Interoperability impacts every participant in a content value chain:

- *Consumers* expect that content that they have legitimately acquired will work with all of their devices, not just a subset. Too often, the opposite is true: content acquired illegally is typically far more portable than electronically distributed content sold through legitimate means. Until this imbalance is addressed, consumers will not fully embrace the electronic distribution of secure media.
- *Distributors* of content are forced to make technology choices that limit the utility of their services to consumers. DRM technology choices adopted by a media distributor are in turn imposed upon their customers, who are forced to choose from a limited subset of all media devices available in the market.
- *Content Providers* have a smaller market footprint and a declining amount of control over the business models under which their content is distributed. The choice of a single DRM technology restricts the business models that can be supported by media distributors, which in turn limits the options available to content providers.
- *Device Manufacturers* are constrained by economic and usability considerations, limiting the number of protection technologies that can be integrated into devices. Device Manufacturers are forced to choose, making the success of devices dependent more upon the success of services that integrate the same DRM technology than on the features or desirability of the devices themselves.

The lack of interoperability leads to a fragmented marketplace and the development of clusters of devices and services aligned only by their use of the same DRM technology. As a result, non-aligned, innovative technologies can be squeezed out of the market, reducing competition and consumer choice.

The Coral Consortium was formed to address the DRM interoperability problems described above by creating a series of technical specifications that allow participants in a content value chain to compete and cooperate in new ways, removing the inhibitions imposed by non-interoperability. The result is not only to permit greater competi-

tion among DRM systems, but to enhance the potential for competition among device makers and service providers as well.

The technical components of the Coral specifications [CCA] include:

- A set of specifications that allow trusted, secure communications between different entities in the value chain.
- An identification of the DRM functionalities that appear in a typical value chain and a set of standardized interfaces between them.
- A strategy for integrating DRM systems with the interoperability framework that imposes minimal (if any) changes on the DRM systems themselves.

The latter aspect of the Coral specification is the focus of this application note, and constitutes one of the primary ways in which Coral differs from other interoperability specifications. Specifically, Coral recognizes that proprietary and strategic interests will continue to exist in the DRM marketplace for the foreseeable future and embraces this diversity while introducing an integration strategy to render the diversity manageable.

This application note describes the way in which the Coral framework can be used with one of the most widely-deployed DRM systems today, Windows Media[®] DRM (WM-DRM)¹. As demonstrated below, integration of Windows Media DRM technology into an interoperable Coral deployment is very similar to integration of WM-DRM into a non-interoperable deployment, and works with the WM-DRM SDKs [WM-DRM] in a straightforward manner, without requiring modifications to the SDKs.

The document begins with a discussion of the Coral approach to interoperability, focusing on those aspects of the architecture that are necessary for understanding how to integrate Coral with DRM systems. The next section contains a discussion of the Windows Media technology itself and an example of how the technology is used in a typical non-interoperable deployment. Finally, the integration of the two technologies is discussed, both for client-side and server-side systems.

2 The Coral Interoperability Model

The Coral interoperability framework is a thin layer of standardized technology that mediates between different DRM technologies in an attempt to provide the type of consistent, predictable experiences that users have come to expect from media formats such

¹ Windows, Windows Media, ActiveX, Internet Explorer, and Microsoft Windows IIS are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

as CD and DVD. In general, DRM systems prevent this type of consistent experience; the goal of the Coral Consortium is to provide technology that can be used to overcome this restriction.

The fundamental questions that the Coral framework attempts to address are:

- Given distinct DRM systems, with their own license expression mechanisms, enforcement schemes, and technology dependencies, how can a user obtain licenses for all of these DRMs with equivalent functionality? In solving this problem, the Coral framework imposes the following additional constraints:
 - The licenses cannot be assumed to be exposed by either DRM in a form that can be directly translated or mapped.
 - The DRM systems may vary in their level of sophistication; there may be some concepts that can be expressed in one DRM that are not capable of being expressed in the other.
 - The interoperability solution must not impose a rights expression or enforcement mechanism on existing systems and must require as few changes as possible to these systems.
- Under what circumstances can systems built and deployed by different manufacturers communicate with one another in a trusted way without proprietary, bilateral relationships?
- What kind of organizing principles govern the transfer of rights among different DRM systems? Are such transfers always possible? If not, how are they restricted?

These three questions and their attendant constraints inform the approach adopted by the Coral framework. The sections below describe various aspects of the technical solution provided by Coral with the intention of providing enough background to understand the way in which Windows Media DRM enabled systems can be made interoperable by integration with the Coral framework.

2.1 Management of Rights

One of the primary assumptions guiding the evolution of the Coral solution to the DRM interoperability problem is the assumption that, in certain circumstances, DRM licenses cannot be inspected. Windows Media DRM is one such case — there is no interface provided by Microsoft that allows WM-DRM adopters to request the license that is actually used to govern any particular piece of content. It is usually possible to obtain some information about the license, but the license data itself is inaccessible.

Even in cases where a license can be obtained and evaluated, it is not clear that mapping such a license to another license expression scheme is always computationally or semantically feasible. For example, how should a license that allows a particular action to occur five times be mapped to a rights expression mechanism that can only restrict the frequency of events to never, once, or always?

As a result, Coral adopts a *license derivation* approach, in which DRM licenses are derived from a common policy artifact known as a *Rights Token*. Handling such Rights Tokens is one of the primary responsibilities of the Coral framework.

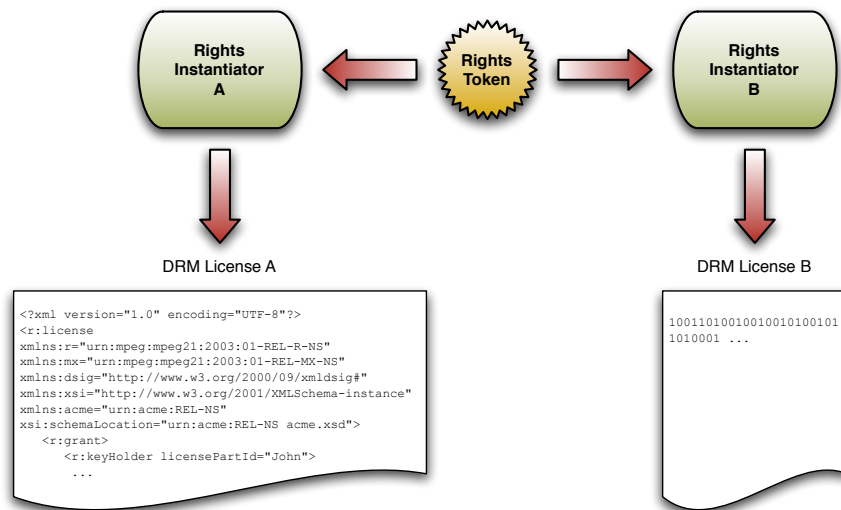


Figure 1: Coral license derivation involves creation of multiple, specific DRM licenses based on a single Rights Token. In this instance, a license expressed in a rich format (MPEG-21 REL) and a proprietary binary license are derived from the same Rights Token. Compliance rules ensure that this process produces “equivalent” licenses from the rights holder’s point of view.

2.1.1 Rights Tokens

Stated briefly, a Rights Token is a data structure with three primary subcomponents:

1. A *principal identifier* that indicates one or more binding targets for the rights. Every principal name is associated with a namespace in which it is meaningful. For example, a user name at a given storefront can be used as a principal in a namespace defined by that storefront. The names themselves need not be interoperable, but the method for expressing them is standardized.
2. A *resource identifier* containing the name of the content resource for which rights

were acquired. Resource names are placed in a namespace context just like principal names, and these namespaces may be unique to a given system.

3. A *usage model identifier*, specifying which rights apply to the given principal and content pair. The usage model identifier is an open data structure that can be as expressive as needs dictate and can vary from system to system.

Rights Tokens provide a way for systems built by different manufacturers to work together by providing a standardized syntax. Note, however, that each of the major subcomponents of a Rights Token may exist in a namespace that is understood only by a single system. How then is it possible to provide interoperability between systems?

2.1.2 Mapping and Standardization

Two techniques are used in the Coral framework to provide interoperability when handling principal identifiers, resource identifiers, and usage model identifiers from different namespaces:

1. *Mapping* — identifiers are mapped across namespaces by known or discoverable relationships among identifiers.
2. *Standardization* — a set of value chain participants chooses a standard set of namespaces for a particular deployment environment.

2.1.2.1 Mapping Between Namespaces

Each of the three subcomponents of a Rights Token can be mapped between namespaces using one of two types of relationship:

1. *Equivalence Relations* — If $A \leftrightarrow B$, then identifier B can be substituted for identifier A , and vice-versa.
2. *Directed Relations* — If $A \rightarrow B$, then identifier B can be substituted for identifier A , but not the other way around.

Examples of how such mappings might be used include:

- An identity federation system can be used to maintain equivalence relations between principal identifiers, so that rights established for a user identified in one system can be mapped to rights for an equivalent user in a second system.
- The same concept may apply to resource identifiers to allow different vendors to use their own numbering scheme for identifying content resources. The ability to

realize such mappings allows Coral to integrate with legacy content distribution systems.

- Different types of identifiers can be mapped to one another via directed relations, which allows, for example, a single identifier for a group of devices to be mapped to distinct identifiers for each member of the group via directed relations. This setup would allow individual device identifiers to be substituted for the group identifier. This type of model is the basis for interoperable authorized domains in Coral.

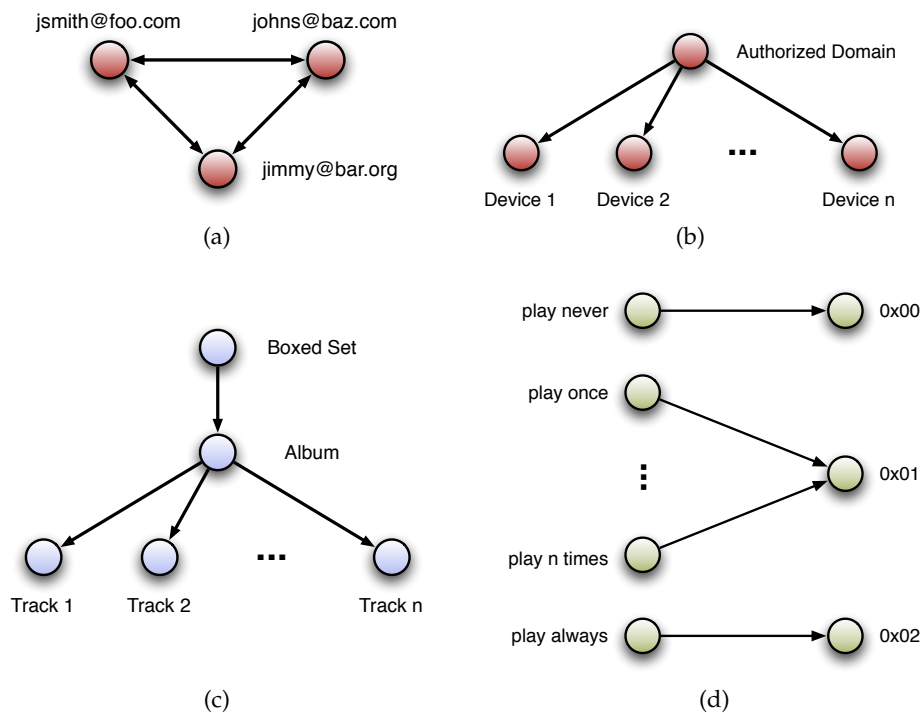


Figure 2: Examples of relations between principal, resource, and usage model identifiers. (a) Equivalence relations between principal identifiers (b) Directed relations between principal identifiers (c) Directed relations between resource identifiers (d) Directed relations between usage model identifiers. The first two relation types are managed by Principal Managers in Coral.

The manner in which various identifiers map to one another is not constrained by the basic Coral framework, but the semantics of these mappings must be understood by all systems that need to manipulate them. The mapping semantics are typically defined in a second specification called a Coral Ecosystem, which adds a layer of specific semantics necessary for a given deployment to the underlying Coral framework.

To extend the authorized domain example above, a Coral Ecosystem might define a type of principal identifier for the group of devices, a type of principal identifier for the devices themselves, and conditions under which relations may be established between them. The Coral Ecosystem would also need to define entities (such as Domain Managers) that maintain and expose the relationships between principal identities.

2.1.2.2 Standardized Namespaces

The second strategy for providing interoperability for identifiers is standardization. No mapping would be necessary, for example, if the namespace and specific usage model identifiers that appear in Rights Tokens were drawn from a standardized set. The Coral framework allows Coral Ecosystems to standardize around namespaces when it makes sense to do so for a given deployment environment. For example:

- The participants in the Ecosystem may choose to use a consistent namespace for numbering resources, such as the Global Release ID system [GRid] for sound recordings.
- Ecosystem participants may choose to label a standard set of usage models with the numbers 1 through n and enforce the rule that each system built to the Ecosystem specification should understand the meaning of the enumerated models.

2.1.3 Deriving Licenses

Rights Tokens are the basic medium of exchange between systems in Coral. Their primary purpose is to record enough information to allow equivalent² licenses for specific DRMs to be derived, providing a consistent usage experience for consumers. The derivation of licenses from Rights Tokens is a distributed process.

The Coral framework defines an entity called a *Rights Instantiator* whose responsibility is to create DRM licenses from Rights Tokens. The primary goal of a Rights Instantiator is to produce DRM licenses that enforce the usages expressed in Rights Tokens. A Rights Instantiator consists of at least three components:

1. A standardized interface called *RegisterRights* by which Rights Tokens are passed to the Rights Instantiator. These Rights Tokens are persistently maintained by the Rights Instantiator.
2. A component that converts stored Rights Tokens into a sequence of interactions with a native DRM license server which ultimately lead to the creation of native DRM licenses that enforce the usages allowed by the Rights Token.

² As judged by the rights holders participating in a given Coral Ecosystem.

3. One or more native DRM license servers that are capable of generating licenses that enforce the usage models standardized or mapped by a Coral Ecosystem. DRM clients acquire licenses from the DRM license server using the protocols defined by the DRM system; these transactions are not defined by Coral.

Many Rights Instantiators may be available for a given Ecosystem, each of which creates licenses for one or more DRM systems. There is no centralized entity responsible for mapping Rights Tokens to DRM licenses. Coral Ecosystem compliance rules ensure that Rights Instantiators produce DRM licenses that enforce the Ecosystem usage models to the satisfaction of the designers of that Ecosystem.

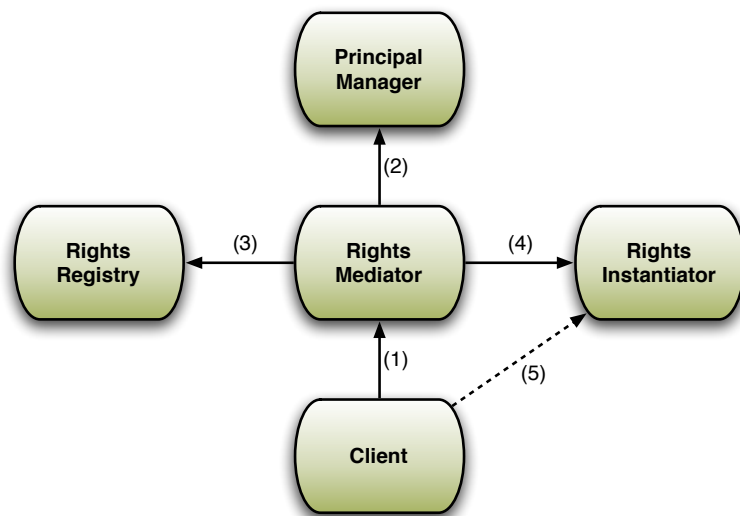


Figure 3: Basic roles in the Coral framework and the interactions between them: (1) a Client initiates the rights mediation process by invoking a Rights Mediator service (2) The Rights Mediator verifies mappings between principals using a Principal Manager (3) The Rights Mediator verifies that rights to derive licenses for a given resource exist at a Rights Registry (4) The Rights Mediator registers the rights at a Rights Instantiator, which realizes those rights as concrete DRM licenses (5) Out of band with respect to Coral, the DRM technology integrated with the Client acquires a DRM license from the DRM server integrated with the Rights Instantiator.

2.2 Rights Mediation

Rights mediation is the most important transaction defined in the Coral framework; it is the process by which rights move between systems integrated with different DRM systems. During rights mediation:

- Rights Tokens are retrieved from entities that store them persistently. These systems are known as *Rights Registries*, and can be characterized as DRM-agnostic rights lockers.
- Mappings between the principal identifiers, content identifiers, and usage model identifiers contained in the Rights Tokens are verified. The systems that maintain these relations, and the type of relations they maintain, are defined by specific Coral Ecosystems, thus making the rights mediation process Ecosystem-dependent. An example of such a system would be a Domain Manager that maintains mappings between an identifier for a set of devices and the identifiers for the devices themselves. Systems such as Domain Managers are instances of a more general type of entity known as a *Principal Manager* defined by the Coral framework.
- Policy decisions are made as to whether sufficient authorization exists to allow the requested movement of rights between systems. This policy decision may depend on factors such as:
 - The presence or absence of particular mappings between principal, resource, and usage model identifiers (e.g. domain membership, valid usage model mappings, etc.)
 - The support of a given destination system for certain usage semantics (e.g. a tamper-resistant time source for subscription models)
 - The number of times such transfers have been allowed in the past
 - The identity of the user behind the request to transfer rights
- Rights Tokens are delivered to Rights Instantiators that will convert them into DRM licenses. Ecosystem-specific compliance rules ensure that the generated licenses enforce the appropriate usage semantics.

These functions are performed by an entity called a *Rights Mediator* in the Coral framework.

2.3 Nodes, Roles, and Interfaces

Standardization of roles and interfaces is essential for interoperability between systems built by different manufacturers. For this reason, Coral defines three key enabling concepts:

1. *Nodes* — secure, named entities that are associated with an asymmetric key pair that is used for secure communications with other nodes. A node is an active entity that plays one or more roles in Coral.

2. *Roles* — units of functionality defined by the Coral specifications or Coral Ecosystems. A node possesses one or more roles that determine how it behaves in interactions with other nodes. In order to hold a given role, a node must exhibit the normative behavior defined for the role and expose the interfaces required of role holders. Several roles have been mentioned in the foregoing discussion, including Rights Mediator, Rights Instantiator, and Principal Manager.
3. *Interfaces* — standardized communication patterns used by nodes with certain roles. The interface specifications define the syntax of request and response messages that are used to accomplish certain tasks defined in the specification.

The primary model for augmenting DRM systems with Coral functionality is by integration of DRM-specific behavior into Coral nodes with particular roles. For example, a DRM license server is integrated into a node with the Rights Instantiator role. This type of integration allows other Coral nodes to interact with the Rights Instantiator using standardized interfaces regardless of the underlying DRM systems with which the Rights Instantiator is integrated. Examples of this type of integration appear in Section 4, below.

3 Windows Media DRM

Windows Media DRM is one of the most widely-deployed content protection systems in existence. A number of factors account for the popularity of WM-DRM:

- *Availability of SDKs* — The popularity of WM-DRM is due in no small part to the availability of WM-DRM technology in the form of Software Development Kits (SDKs) that enable a wide range of adopters to integrate the technologies into their larger offerings. For example, a media distributor operating a web-based storefront can license and integrate the WM-DRM License Server without undertaking a massive new development project.
- *Approval of Content Companies* — WM-DRM is a well-known technology among content providers. Major content providers understand the security of the system and have allowed their content to be distributed with WM-DRM protection. As such, content resellers that choose to rely on WM-DRM can likely avoid issues of security and robustness of the DRM that might otherwise slow down negotiations with content providers.
- *The Windows Media Platform* — The market success of the Windows operating system means that WM-DRM will continue to be one of the most widely-deployed

DRM systems and make WM-DRM an important part of any interoperability strategy for content distributors.

- *Compatible Devices* — Microsoft also licenses a device SDK to OEMs, which has led to the emergence of a robust market for WM-DRM enabled clients that interact naturally with the Windows Media software platform. The ready availability of devices further strengthens the argument for adopting Windows Media.

The factors adduced above make adoption of WM-DRM a compelling proposition in many circumstances. As with any single-vendor technology, however, there are attendant disadvantages. Foremost among these is a lack of interoperability with other content protection formats that may be preferred by other members of the content distribution value chain for various reasons. For example, reliance on WM-DRM to the exclusion of other content protection systems limits customers of any particular content distributor to a subset of available devices.

Section 4 will demonstrate that the availability of SDKs for Windows Media DRM enables the use of WM-DRM technology within a larger ecosystem of content protection systems, all interoperating via Coral.

3.1 Windows Media DRM SDKs

Windows Media DRM is an evolving technology. Over time, the various SDKs that make up WM-DRM have been updated to add features or incorporate fixes. At present, there are four primary WM-DRM SDKs as shown below. As new versions of Windows Media become available, these SDKs will likely be superseded.

- *Windows Media Rights Manager* — This is the primary SDK for constructing WM-DRM enabled systems. In particular, it includes content packaging and license generation components that allow content distribution adopters to build license servers and offer their content in WM-DRM protected formats.
- *Windows Media Format* — The Windows Media Format SDK handles Windows Media content formats at the consumption side, including DRM-protected content. This SDK provides the primary means by which clients interact with Windows Media enabled services. This SDK also contains Device Manager technology that governs the interaction of a PC client with portable devices.
- *Windows Media DRM for Portable Devices* — This SDK is intended for use in devices such as portable audio and video players, and provides that subset of the functionality of the Format SDK necessary for enforcement of governance in the portable device environment. Portable device clients built using this SDK are not

fully general clients like those built on the Format SDK; they must be tethered to a more sophisticated client for full functionality.

- *Windows Media DRM for Network Devices* — This SDK allows devices to access content formatted in Windows Media remotely over a network. The Network Devices SDK is not intended for use in devices that store DRM-protected content persistently; such devices integrate with the Portable Devices SDK, above.

3.2 Deploying Windows Media DRM

The Windows Media SDKs described above provide adopters with the tools necessary to build secure content distribution systems, but do not in themselves provide a ready-made solution. A certain amount of integration work is required to incorporate WM-DRM into a complete content distribution system. This section describes the integration work that must be undertaken by various parties in order to adopt WM-DRM.

Before describing the details of integration with the Windows Media SDKs, it is important to make a clear distinction between those aspects of the problem that are handled by the Microsoft SDKs and those parts of the problem that must be solved by the integrator. This discussion is followed by a description of the various methods that can be used at the client and server sides to perform the technical integration with WM-DRM.

3.2.1 Division of Labor

The Windows Media SDKs implement a set of proprietary protocols that cannot be modified by integrators. These protocols are not published by Microsoft, so adopters are required to use the WM-DRM SDKs to perform certain tasks involved in distributing secure content. The tasks handled by Windows Media SDKs include:

- Requesting a license from the client side
- Producing the license for a given client at the server side
- Persistently storing the license at the client side
- Evaluating the license when accessing the protected content
- Requesting revocation of a license from the client side

The tasks performed by Windows Media, described above, cannot be changed by adopters. However, there are necessary aspects of the protected content acquisition and consumption cycle that are not handled by Windows Media, and thus left entirely to integrators.

1. *Why seek a license?* The WM-DRM SDKs do not specify the conditions under which a client will try to obtain a license for a given piece of content. The decision to obtain a license is usually, but not necessarily, driven by a user interaction. When a user attempts to acquire a new piece of content, or to perform a governed action on protected content that has already been acquired, the integrated client application must invoke the appropriate interfaces to trigger WM-DRM to seek a license for the content in question. In the interoperable environment defined by Coral, the impetus to seek a WM-DRM license may come as a result of an interoperability transaction.
2. *Why grant a license?* Significantly, the Windows Media SDKs do not dictate the conditions under which a license request should be granted. In a typical scenario, a server-side process will attempt to correlate a license acquisition request with a previous transaction in which a user acquired the rights to acquire a license. For example, when a user purchases content from a web-based storefront, the service may generate a transaction record with a unique serial number that is included in the license request. Upon receiving such a request, the storefront can use the serial number to check that the rights were legitimately acquired and then generate an appropriate license.
3. *What kind of license should be generated?* Windows Media DRM does not restrict the kind of licenses that can be generated for a given piece of content; the precise nature of the license generated is up to the content provider and content distributor, based on commercial arrangements with their customer. The licenses generated are limited only by the expressibility of Windows Media DRM itself.
4. *Why evaluate a license?* There are no limitations on evaluating a license once acquired, except for limitations that are expressed in the license itself. That is, the WM-DRM SDKs do not provide a trigger or any decision logic around the process of deciding to access content. In a typical deployment, license evaluation might be triggered by a user action such as double-clicking on the content name in a list.
5. *Why seek to revoke a license?* WM-DRM provides a limited means for revoking issued DRM licenses. The revocation process must be initiated by the client side; there are no means to allow a server to unilaterally revoke an issued license. An important integration task is to implement the mechanism by which a client is signaled to seek a license revocation.

To summarize the discussion above, Windows Media DRM SDKs answer the questions as to *how* certain tasks are accomplished, but give no indication as to *why* these tasks would be requested in the first place. This design gives adopters the flexibility to implement a wide variety of business models on top of WM-DRM, and even enables

WM-DRM to be incorporated into an interoperability framework, as described in Section 4, below.

3.2.2 Client Side Integration

Content protected with Windows Media DRM is ultimately consumed by a rendering client. Therefore the first task of a rendering client is to obtain WM-DRM protected content and a corresponding license that can be evaluated to grant access to that content. Windows Media DRM has traditionally made a technical distinction between PC and non-PC devices in license acquisition — licenses for portable and network devices needed to be generated based on licenses obtained for a PC. For this reason, this section discusses acquisition of content and licenses via a PC client built on the Windows Media Format SDK.

Clients typically integrate with the WM-DRM portion of the Windows Media Format SDK in one of the following two ways:

- *Web-based integration* — This type of deployment is very common for Windows-based PC clients that will acquire Windows Media DRM protected content from a web-based storefront. The basis for this approach is the delivery (in a web page served by the storefront) of parameters that trigger an ActiveX control called RMGetLicense. This ActiveX control initiates the acquisition of a WM-DRM license and content.

This approach makes several assumptions about the interacting technologies:

- The web browser is Internet Explorer
- The client is running a version of the Windows operating system
- ActiveX components are installed and instantiated on demand
- The storefront website is being served by Microsoft Windows IIS

Despite these restrictions, the web-based integration approach remains one of the most popular ways to deploy WM-DRM.

- *Programmatic integration* — The web-based integration approach described above provides a convenient mechanism for acquiring WM-DRM licenses via the web. It is possible, however, to build custom clients using the Windows Media Format SDK that interact directly with WM-DRM without the intermediary of an ActiveX control. In fact, the ActiveX control is simply a convenient shortcut that integrates a web-based user interface with the rest of the client components. Direct interaction with the Format SDK is the most likely type of integration for custom clients incorporating WM-DRM. The programmatic approach makes fewer assumptions

about the presence of supporting Microsoft technologies deployed at both the client and server sides.

The walkthrough scenario in Section 3.2.4 below shows the web-based integration scenario, whereas the Coral-integrated scenario in 4 relies on a programmatic integration.

In both of the cases described above, the designer of the integrated system must determine *why* a given client should seek to acquire a WM-DRM license. The integration scenarios differ in *where* the decision to acquire a license is made: at the server side in the web-based approach or at the client side in the programmatic approach.

3.2.3 Service Side Integration

There are two primary technical integration tasks to be performed at the service side:

- *Correlate acquisition of rights with license generation.* In a typical deployment, the service would integrate a commerce front-end with a database that records the details of transactions, such as:
 - A unique serial number for the transaction.
 - A record of the user account that was used to acquire the rights.
 - Identifiers for the content for which rights were acquired.
 - Some indication of what kind of rights were acquired. For example, were the rights to the content obtained as part of a subscription? How many clients can request licenses based on the same purchase? Etc.
 - Other details pertaining to the transaction, such as payment information.

The unique transaction number, if included in a license request, can be used to retrieve the details of the transaction. On the basis of these details, the service can decide how to ask the WM-DRM license server to generate a license with the appropriate permissions. This is one of many different ways to correlate the license being requested with the transaction that authorized the license acquisition.

In the Coral integration scenario described in 4, the record of which rights have been acquired is provided by an interface standardized in the Coral specifications.

- *Implement decision logic for license requests.* The Windows Media DRM SDKs provide hooks that allow service integrators to insert custom decision logic in the chain between the license request and the generation of the DRM license. Technically, license requests are made by clients using an object that Microsoft calls a *Challenge Object*. The Challenge Object is used to request a license, but integrators can also insert headers that can be processed by custom decision logic at the server side.

To extend the correlation example, a client might include the transaction serial number as a header in the Challenge Object. This header would be evaluated by the custom decision logic before passing the Challenge Object on to the WM-DRM license generation process. Another way to solve this problem would be to insert content and user identifiers into the challenge object, which the decision logic would then use to look up the acquired rights for the given pairing of user and content.

The custom decision logic decides whether or not to forward the Challenge Object to the standard WM-DRM license generation processes. If it decides to do so, it may modify the Challenge Object in order to influence the licenses that are eventually generated. Once the license is generated, it is returned to the requester as an opaque object and cannot be modified.

3.2.4 A Windows Media Walkthrough

This section describes a typical license acquisition scenario based on acquisition of content rights via a web storefront. This scenario forms the basis for the Coral integration scenario described in Section 4 this document.

This scenario assumes that the storefront is served by the Microsoft Windows IIS web server and that the user is purchasing content via Internet Explorer from a PC. The sequence of events is illustrated in Figure 4.

1. The user acquires rights to a piece of content via a commerce interface. This step bundles together all issues relating to authenticating the user, selecting the content, and processing the payment information. In reality, each of these would likely be separate steps. For the purposes of this Application Note, however, the individual steps are not significant.
2. The web application writes a record to a transaction database, indicating which rights were acquired for which content, and by which user. These data are retrieved later as part of the decision logic in step (7).
3. A confirmation web page is returned to the user, containing embedded parameters for the instantiation of an ActiveX control.
4. The ActiveX control is executed with the parameters returned above, which triggers interaction with the WM-DRM Format SDK on the client side.
5. The WM-DRM component generates a Challenge Object and sends it to a web application hosted at the server side. The Challenge Object contains sufficient information to identify the rights acquisition transaction that occurred as part of

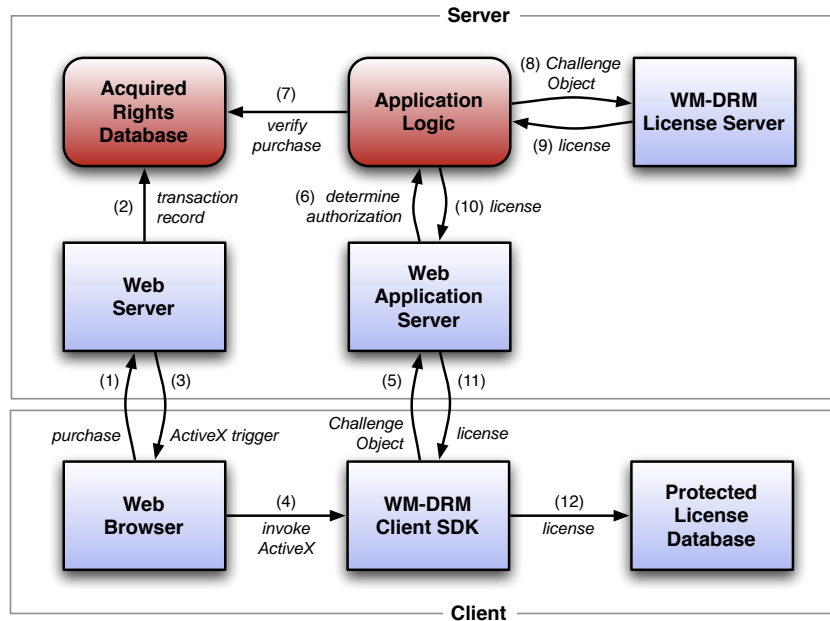


Figure 4: Integrating Windows Media DRM with a web storefront.

- step (1). The transaction between the client and server side components uses a proprietary Microsoft protocol.
6. The web application server, after verifying the validity of the DRM client component, passes the Challenge object on to custom application logic deployed by the system integrator.
 7. The application logic retrieves the details of the content acquisition transaction using information contained in the Challenge Object.
 8. The application logic takes a decision based on the Challenge Object and the details of the rights acquisition transaction. In this case, it decides to continue with a license request. The application logic modifies the Challenge Object to ensure that a license with the appropriate parameters is generated, then passes the Challenge Object on to the license server, which will generate the license.
 9. The license server logic generates the WM-DRM license for the requesting client and returns it as an opaque object to the custom application logic.
 10. The custom application logic returns, passing the license object back to the application server.

11. The web application server returns the license to the client.
12. The client stores the license in a protected database, using a scheme proprietary to WM-DRM.

The next describes how this walkthrough changes when a Windows Media client and service are integrated into the Coral DRM interoperability framework.

4 Integrating Coral and Windows Media

This section describes the integration of Coral and Windows Media DRM at both the client and service. These examples extend the Windows Media DRM walkthrough of Section 3.2.4, but describe how clients and services based on WM-DRM can cooperate with other Coral participants using the standard mechanisms described above.

The Coral framework divides the acquisition of content into two distinct phases that are not normally separated in systems built around a single DRM technology:

1. *Acquisition of Rights* — In this step, a consumer acquires the right to obtain one or more instances of a content resource. For example, the right to acquire differently-formatted versions of the same legitimately purchased song for all devices in a given set. In Coral, this right is represented by a Rights Token.
2. *Fulfillment of Rights* — The fulfillment step involves the creation of one or more concrete DRM licenses based upon the right acquired in the first step. This step is performed in Coral by deriving DRM licenses from Rights Tokens.

In non-interoperable system, these two steps are almost always performed by the same business entity and are often inseparable. To provide interoperability between services, however, it is necessary to be able to separate the two steps. For example, a single commercial entity that offers interoperable rights may be able to fulfill the rights in one DRM format (such as WM-DRM), and partner with other entities that provide licenses in other DRM formats³. Other scenarios are possible as well. The Coral framework provides a standardized means by which rights obtained from the point of acquisition can be conveyed to and used by other participants.

³ It is also possible for a single vendor to use the Coral framework to provide DRM interoperability purely within the context of a single service.

4.1 Fulfilling Rights Acquired Elsewhere

In this example, Rights Tokens that were acquired from another point of purchase are registered with a WM-DRM-based service and are fulfilled in the form of WM-DRM licenses. This example shows one aspect of integration with Coral: the ability to receive and fulfill rights obtained from another Coral participant.

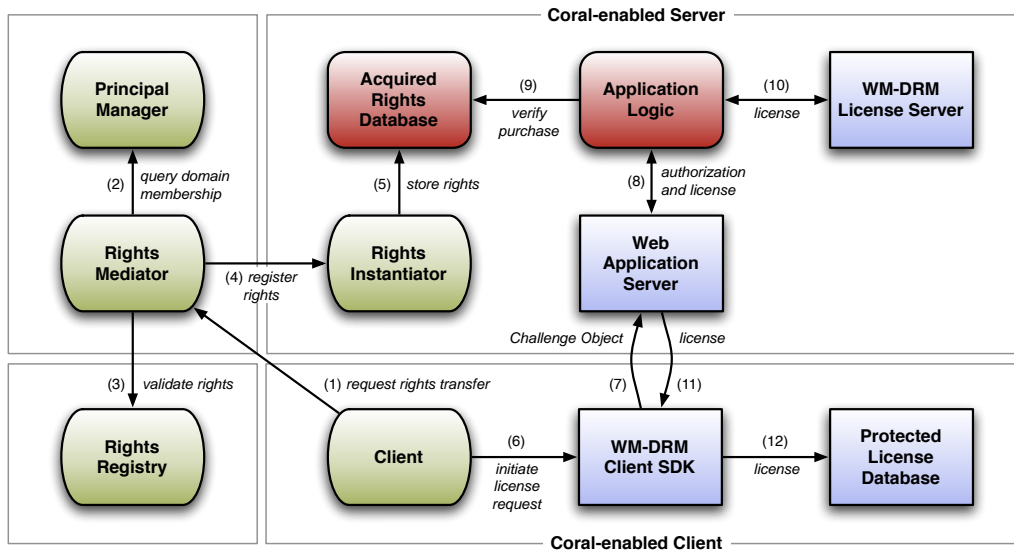


Figure 5: Integrating Windows Media DRM with Coral. In this example, rights acquired elsewhere are made available to a WM-DRM-based server and are fulfilled in the form of a WM-DRM license.

Figure 5 illustrates the scenario, with the following graphical conventions:

- Each grey box represents a node. Note that some of the nodes contain only Coral roles while others are integrated with Windows Media technology.
- Every node has one or more Coral roles, indicated by rounded green boxes.
- Square blue boxes represent functionality provided by Microsoft as part of the WM-DRM SDKs or the IIS web server.
- Rounded red boxes represent non-Coral points of integration that serve as intermediaries between Coral and Microsoft functionality.

4.1.1 Walkthrough

The specific actions taken at each step in the example of Figure 5 are described below. For more specific information about the WM-DRM components, see [WM-LG].

1. A node containing a Coral Client requests rights from a Rights Mediator.
2. The Rights Mediator invokes a Principal Manager to determine whether or not the Client is authorized to obtain the rights. To take a specific example, this Principal Manager may be acting in a certain Ecosystem as a Domain Manager, reporting the domain membership status of the requesting Client.
3. The Rights Mediator verifies that the rights that are being requested actually exist by querying a Rights Registry. A specific example would involve checking with a second Coral-enabled storefront to ensure that an appropriate Rights Token exists for the resource being requested.
4. Upon making a successful rights mediation decision, the Rights Mediator creates a new Rights Token containing the appropriate identifiers and registers this Rights Token with a Rights Instantiator. The Rights Instantiator returns an opaque trigger that ultimately provides the Client with enough information to seek a Windows Media license for the requested resource.
5. The Rights Instantiator stores a representation of the Rights Token in a database of acquired rights. This step is identical to step (2) in Section 3.2.4, except that the input come from an external system rather than an internal process triggered by a purchase event.
6. The Client, having received a license trigger that indicates the server from which it should request a license, signals the WM-DRM SDK at the client side to initiate the license acquisition.
7. The WM-DRM SDK at the Client side sends a Challenge Object to the license server. The Challenge Object contains enough information to allow the license server to correlate the license request with the rights registration of step (4).
This step is identical to step (5) in Section 3.2.4. All of the following steps in this transaction mirror steps in the previous Windows Media walkthrough.
8. After verifying that the requesting entity is a legitimate Windows Media client, the web application server forwards the Challenge Object to custom application logic for further processing.
9. The custom application logic correlates the information in the Challenge Object with the rights that were registered in step (4).

10. After verifying that the Challenge Object actually represents acquired rights, the application logic forwards the Challenge Object to the license generator, along with keying material and a *WMMRights Object* that specifies which rights are to be granted in the generated license.

In the Coral model, this application logic is responsible for mapping the usage model identifier contained in the Rights Token that was registered in step (4) to specific rights expressible in WM-DRM. The nature of this mapping is Ecosystem-dependent, but a simple example will serve to illustrate the process.

Suppose that the Ecosystem in which the server is operating defines a usage model identifier structure that consists of a usage model number followed by a prescribed number of parameters. For example, an Ecosystem could define the usage model identifier:

```
<eco-model number="3" expiration="2007-01-31T13:20:00.000-07:00"/>
```

All systems based on this particular Coral Ecosystem understand that usage model 3 is to be interpreted as an unrestricted play model, with a defined expiration date. Such a model might be used, for example, in a rental scenario.

The application logic would turn this usage model identifier into a *WMMRights Object* by setting the parameters *WMMRights.AllowPlay* to *true* and *WMMRights.ExpirationDate* to *#20070113 20:20:00Z#*, in accordance with the format specified in the WM-DRM SDK. These settings create the appropriate conditions in the WM-DRM license to enforce the Ecosystem-specific usage model identified in the registered Rights Token.

It is important to note that the keys used in the generated license are created by the content packager and not any Coral-defined system. Content key management operates in accordance with the procedures defined by the DRM, without interference from the Coral framework. Interjecting non-DRM systems into content key management would violate the compartmentalization that the Coral integration-based model makes possible, and would introduce an unnecessary security risk. More details on WM-DRM key generation can be found at [WM-KEY].

11. The newly-generated license is passed back to the requesting client by the web application server.
12. The license is placed in a protected database under the control of WM-DRM and cannot be manipulated further until it is evaluated or revoked.

4.1.2 What Has Changed?

A comparison between Figures 4 and 5 shows the major differences between a standard integration of Windows Media DRM into a web-based storefront and incorporation of the Coral interoperability framework into the same environment. The primary technical change is the replacement of the web-based user interface components with standardized, Coral-based components that provide an alternative means for requesting and registering rights at the server. The other components are very similar or identical to the components in the Windows Media-only deployment described in Section 3.2.4. The WM-DRM SDKs are not modified for the new scenario.

As noted in Section 3.2.1, several important questions must be address when performing any integration with Windows Media DRM. The fundamental change introduced by the Coral framework is the need to answer some of those questions in a different way:

1. *Why seek a license?* In the pure WM-DRM walkthrough in Section 3.2.4, a client seeks a license because it has acquired the right to do so by purchasing the content from a web-based storefront. In this example, the client's motivation is slightly more sophisticated: it seeks a license because it has obtained the right to acquire a license by purchase from a different location.
2. *Why grant a license?* In Section 3.2.4, the service granted a license because a previous purchase event had caused transaction information to be inserted into a transaction database. The presence of a valid, non-fulfilled transaction ID in the license request Challenge Object indicates to the server that it should allow the license to be generated. In the present example, the transaction database was populated via a different mechanism — the registration of a Rights Token via the Coral RegisterRights interface.
3. *What kind of license should be generated?* The type of license generated in this example depends on the usage model identifiers contained in the registered Rights Token, whose interpretation may be constrained by a Coral Ecosystem specification and its associated compliance rules. In Section 3.2.4, the licenses generated were determined solely by the vendor without consultation or reference to any other entity. It is important to note that Coral Ecosystems are not required to standardize around a set of usage models, but it is likely that many will opt to do so in order to provide consistency of usage to consumers.

4.2 Providing Rights to Others

The previous example showed how rights at an external point of acquisition can be registered at a Windows Media-based service using Coral and subsequently instantiated

in WM-DRM format. Conversely, interoperability requires the ability to provide a representation of rights to other services that can realize those rights in different DRM systems.

Figure 6 shows how a service based on Windows Media DRM provides information about rights to other Coral participants. In this example, the WM-DRM service implements the Coral *Rights Registry* role, effectively taking the place of the system queried in step (3) of the example in Section 4.1.

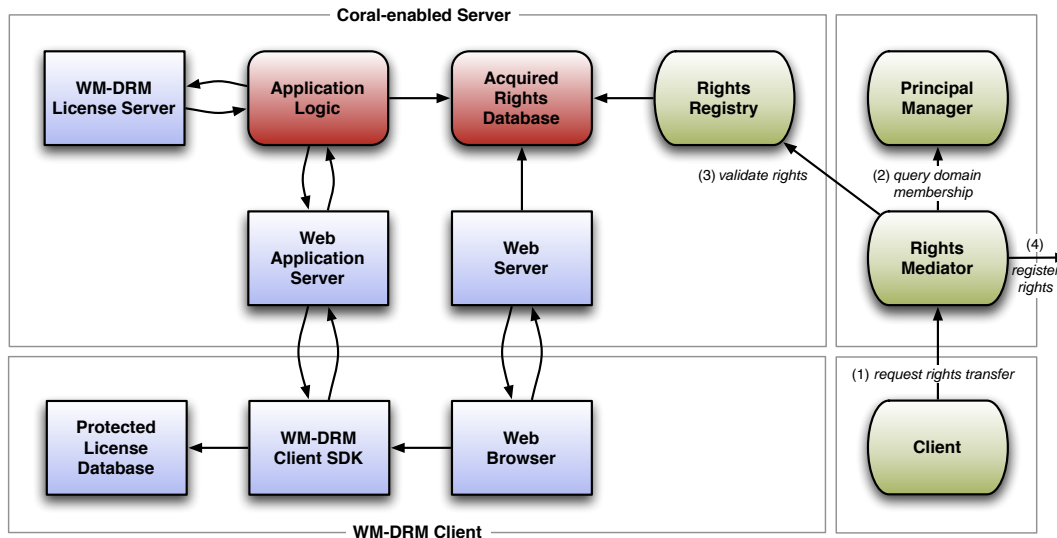


Figure 6: Integrating Windows Media DRM with Coral. In this example, rights acquired elsewhere are made available to a WM-DRM-based server and are fulfilled in the form of a WM-DRM license.

0. In this example, the initial rights are acquired and fulfilled purely within WM-DRM as in Section 3.2.4.
1. A node with the Coral Client role (possibly integrated with a second DRM system) requests rights for another device.
2. The Rights Mediator determines whether or not the new client is authorized to receive the rights, based on the existing relations and parameters defined by a Coral Ecosystem.
3. The Rights Mediator queries the WM-DRM service using the *QueryRights* interface, which returns a Rights Token representing the rights that were first acquired at the WM-DRM service. The integrated WM-DRM service implements a thin

layer of technology that (a) converts the query into a search of its internal transaction records and (b) returns information about the rights acquired in the form of a Coral rights-token.

4. Assuming that the transfer is allowed, the Rights Mediator registers the Rights Token with a second Rights Registry, not shown in Figure 6.

This example illustrates several further aspects of the Coral framework:

- The acquisition and initial fulfillment of rights can occur at a single service, upholding the *status quo* in electronic commerce distribution.
- As such services decide to work together with external entities, they can use Coral to interoperate via trusted interfaces that are internally integrated with their existing business processes.
- The WM-DRM client in this example did not include any Coral technology; the Coral integration appeared at the service side only. The Coral Client that requested the rights transfer may have been integrated with a second DRM client, but this is not required by Coral in general. This is possible because Coral does not inject itself into transactions between native DRM license servers and their clients.

5 Conclusions

This application note has described methods for integrating the Coral interoperability framework into clients and servers based on Windows Media DRM, using the standard SDKs provided by Microsoft. Such integration gives WM-DRM systems the ability to interoperate with non Windows-based devices and services, providing a necessary building block for transparency and predictability of content usage across DRMs. At a higher level, this document has provided insight into the architecture of the Coral interoperability framework and shown a concrete example of how Coral can be incorporated in DRM-specific deployments without interfering with the basic protocols and key management of those DRM systems.

6 References

- [CCA] The Coral Consortium, *Coral Core Architecture Specification v3.0*, July 2006.
<http://www.coral-interop.org>.
- [WM-DRM] Microsoft Corporation, *Digital Rights Management (DRM)*, <http://www.microsoft.com/windows/windowsmedia/forpros/drm/default.aspx>
- [GRid] IFPI, *Global Release Identifier*, <http://www.ifpi.org/grid>.
- [WM-LG] Microsoft Corporation, *Generating and Issuing Licenses*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmrm10/htm/wmrm_sdk_guide_ibrr.asp.
- [WM-KEY] Microsoft Corporation, *WORMKeys Object*, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmrm10/htm/iwmrmkeysobject.asp>.