The Need for a Rights Language

Technical White Paper

Version 1.0

CONTENTGUARD™

Table of Contents

Introduction

The Internet has spawned a revolution in how content is distributed and services are accessed. Industries that engage in the trade of Intellectual Property as well as those that simply generate content and services for use or sale are increasing their dependence on network delivery.

This reality, coupled with the failure of many on-line business models[1] and the increasing availability of broadband services, has fueled the development of technologies to manage, secure, control, and automate the flow of content and the access to services over the Internet. Internal content distribution, external content distribution, retail content for sale, and "on-line services" now depend on the Internet to establish cost effective, reliable, flexible, highly available, and secure means of managing the delivery of these "assets" that are the cornerstone of digital commerce and enterprise communication.

Digital Rights Management (DRM) is the common term collectively associated with such technologies, and the associated workflow is called a "DRM" or "Rights Enabled" workflow. A whole new industry is forming around DRM as an emerging technology. The workflows associated with it are finding their way into complementary technologies such as Digital Asset Management, Content Management, and Trust Systems.

If we consider this lifecycle or workflow for digital content and services, we see that the exchange of rights information is required between the players or entities in the workflow or at each step of the lifecycle. For example, a content user needs to know what rights or permissions are associated with a piece of content. A content distributor or publisher needs to communicate the rights that are available for consumption. An automated content aggregation system needs to know if a piece of content can be published in physical and/or digital format

We also realize that expressing rights can be simple or very complex. For example, a user may obtain the rights for unlimited play for a music file; a corporate document may have the usage right restricted to certain managerial levels, PCs and/or certain dates. Rights expressions get more complex when we try to mimic the use and distributions of content in the physical world. For example, specifying the rights that govern the lending of a digital book or the giving-away of an article in an electronic magazine can be very complex.

Furthermore, the current commercial workflow involves distributors or middle entities for almost everything a consumer needs. Specifying the rights for the different entities in this multi-tier workflow is also needed. For example, the usage rights for a piece of content will change as it moves from the creator, aggregator, distributor, retailer, and consumer.

In an end-to-end system[2], other considerations such as authentication and security become important. For example, you must specify the devices, issuers or users, and the mechanisms to authenticate those entities. A rights specification may be

[1] Such as advertisement supported business models.

[2] And end-to-end system is, for example, a system that delivers protected digital content or an electronic library where registered users can check out electronics books

accessed or manipulated by different participants during different stages of its life cycle, and mechanisms and semantics are needed to validate the authenticity and integrity of the rights expression.

Thus, a common Rights Language that can be shared among the participants in this digital workflow is required. Not only from an obvious interoperability point of view, but more so to comprehend that rights will be manipulated and changed during the digital workflow and lifecycle, and to comprehend system issues such as trust and authentication.

This paper will focus on some key requirements for a rights language and then describe how XrML meets those requirements and the advantages XrML can provide in building "rights enabled" systems.

Requirements for a Language that Describes Rights and Conditions

As DRM technologies are developed to support a wide variety of business model and content formats, the language supporting the DRM must have wide appeal. Namely, the language must be:

- *Comprehensive*: a language capable of expressing simple and complex rights expressions in any stage in a workflow, lifecycle or business model.
- *Generic*: A language capable of describing rights for any type of digital content or service (an ebook, a file system, a video, or a piece of software).
- *Precise*: a language that communicates precise meaning to all the players in the system.
- *System Interoperable*: a language that comprehends that it a part of a tightly integrated end-to-end system. A language shall support those elements that are required for components to interoperate within the context of an end-to-end DRM system. Authentication and validation of entities, open and standard identification systems, security (including integrity and confidentiality) of the rights expressions themselves, comprehension of emerging web technologies, and machine-processability are part of this requirement.

The first three requirements relate to how well the language communicates arbitrary rights associated with arbitrary business models. The fourth requirement emerges when the language in put into action. System interoperability requirements can be thought of as practical or real-life requirements that have been developed from field and implementation experience.

XrML, eXtensible Rights Markup Language

XrML is a language to specify rights. XrML is an XML-based usage grammar for specifying rights and conditions to control the access to digital content and services. XrML had its roots in Xerox Palo Alto Research Center. Digital Property Rights

Language (DPRL) was first introduced in 1996. DPRL became XrML when the meta-language (used to construct the language) was changed from a lisp-style meta-language to XML in 1999.

Using XrML, anyone owning or distributing digital resources (such as content, services, or software applications) can identify the parties allowed to use those resources, the rights available to those parties, and the terms and conditions under which those rights may be exercised.

Since its inception, the language has evolved through industry feedback, critical review, and product implementation. The language has become *comprehensive* by providing a framework to express rights at different stages of a workflow or lifecycle, *generic* by defining a large body of format and business neutral terms (about 100) and using these terms to specify rights to any digital content and service, and *precise* through the development of a grammar and processing rules that enable unique interpretation of the language. XrML is by far the most advanced and mature rights language in use today. Since 1999, the emphasis has been to get the language implemented in real life systems. This experience has resulted in additional system-related features (trust, for example) that are now part of the language. The current version of the language is 2.0.

Why XrML is the Language of Choice

The concepts explained in this paper and the proliferation of tens or even hundreds of XML based languages to share information among systems have validated the need for language that is used to specify and communicate rights. But which rights language? XrML is the choice for a rights language, because it goes beyond mere semantics or syntax concerns. XrML provides the following four main advantages:

- *Open Standards*: XrML is intended to be an open standard activity where industry members can collaborate and contribute their expertise to the language.
- *Useful for Any Business Model*: XrML can be used for many different business models and comprehends multi-tier models.
- *Interoperable*: XrML provides syntactic, semantic, and system interoperability. This interoperability enables XrML to be used as part of a bigger system and comprehends other things such as security.
- *Extensible*: XrML has leveraged open mechanisms to extend the language with new terms. As the industry evolves, there will be activities to standardize terms, create new terms and new business models. XrML has been designed with mechanisms to easily incorporate those terms.
- *Companion Language SDK*: XrML as been implemented in a software development kit (SDK). Developers using the SDK will enable their applications with capabilities to understand and manipulate rights expressions.

Open Standard

The nature of the DRM market demands that the rights language be an open standard. Because DRM encompasses multiple systems spanning the entire life cycle of a digital item, an application at one end (for example, a media player –the consumption end) must understand the rights that have been assigned at the other end (for example, an authoring application -the creation end). It would be difficult for an application to understand multiple incompatible ways to specify rights.

A language that is an open standard also enables the selection of best of breed components, interoperability across business models, and help for users to optimize their implementation and costs. In turn, this reduces barriers to adoption and accelerates growth. It also enables large-scale cooperation to further the capability of the language as it is applied to digital rights management and other related technologies. XrML will be such an open standard.

Useful for Any Business Model

XrML is built on a business neutral model. The terms used to construct the rights expressions (rights, conditions, and obligations) are, to a large extent, business neutral. This allows the use of the language to express rights for many usage and business models in diverse industries.

Rights expressions consist of rights[3] (permissions), conditions, and obligations associated according to a grammar. Rights are modulated by conditions. For example, time conditions can modulate or restrict a view ("view for 30 minutes") right, or a fee condition can be a prerequisite to obtain rights ("$10 to get the permission to print one copy"). The conditions can be made elaborate and comprehensive through a mechanism called "pattern matching". Pattern matching enables the matching of expressions that are used as conditions, but not defined in the language. For example, pattern matching can be used to specify users in a domain ("view permission to anyone who's association matches www.contentguard.com").

XrML applies to markets beyond the content arena. XrML can be used for services, certificates, contracts, software, disk access, shared space rules, and many other arenas. This broad scope also makes it possible to leverage rights-related applications such as enforcement mechanisms (applications that would enforce a certain rights condition such as the number of times a consumer can access a particular content or service) from one area into another.

[3] Although "rights" can have a strict legal meaning, we have chosen –in this paper- to use a more relaxed definition that can be used interchangeably with "permissions"

Examples of different business scenarios

- *Distance Learning*: A digital video lecture at a college is limited to students registered for the course, each of whom was issued a digital certificate identifying them as registrants. Non-registered students may view the course for a metered fee of $10 per hour during the course period.
- *Retail Sale of Content*: www.foocontentseller.com offers to sell digital books with the following terms: unlimited view for $10, unlimited view and print for $15, or metered view at $2.50 per hour. Typically, the customer selects one of these options and the corresponding license is generated.
- *Corporate Product Information*: Corporation, Inc. wishes to distribute information for an upcoming new product. The document is emailed to the sales force. The document is now distributed to remote desktops and laptops. The document can be accessed only if the user is logged-in to the corporate intranet. If the user is not connected, the document is inaccessible.
- *Customer acquisition:* Securities Firm A wishes to use its research reports to gain new customers by distributing them over the Internet. It reports (e.g., published report or video clip of analyst presentation) is available for download if the reader registers information with the firm. As the research is passed from the first reader to his friends and colleagues, is will only be readable or viewable if the recipient goes back to the Firm's website to obtain a license by registering.

Multi-tier business models

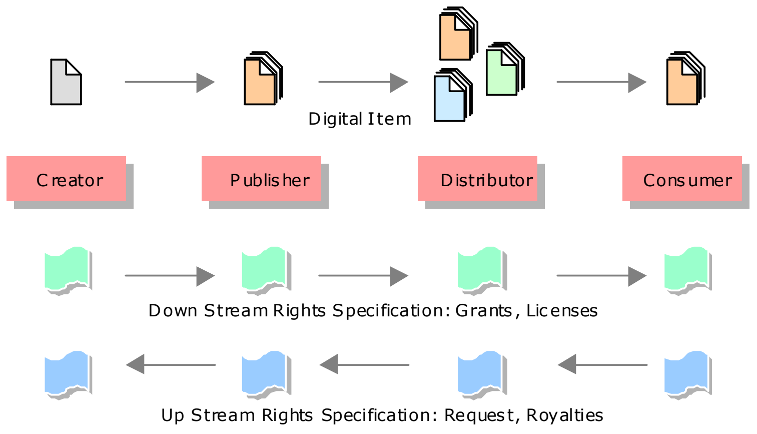
To some, DRM becomes valuable when the rights can be specified across multiple tiers. The rights specified "upstream" control the rights that can be specified "downstream". For example, a publisher specifies the kinds of rights a particular distributor can issue, and once the rights to a consumer are issued, the kinds of rights that an end-user can transfer to a peer. In other words, the rights at every tier are not and cannot be arbitrarily or independently specified.

The XrML language fully comprehends multi-tier business models and provides a mathematically sound way to specify options at each distribution level. In this scenario, XrML can support models such as: publisher to distributor rights specification (e.g., *issue* –as in issue the right to issue a usage license), distributor to consumer rights specification (e.g., *play*, *print*), and peer-to-peer rights specification (e.g., *loan*, *transfer*). Entities can specify not only the rights to the next entity, but also exactly how and when the rights are transformed in the downstream entities.

And within a tier, different models can be supported: retail (*sell*), library (*loan*), and enterprise (*copy*, site license).

Another way to understand this multi-tier model is by viewing a life cycle. Say, for example, a lifecycle for a digital item consists of the following stages: creation, aggregation/publication, distribution, consumption, and super-distribution.

For the creation stage, XrML can specify appropriate rights, such as *create*[4]. For the aggregation stage, XrML can express rights applicable to a process or workflow that aggregates content, such as *edit* and *extract*. For the distribution stage, XrML provides meta-rights, which define rights to content distributors, such as the right to issue rights. For the consumption stage, XrML defines content consumption rights, such as *play* and *print*. For the super-distribution[5] stage, XrML defines rights that control the distribution of content from peer to peer, such as *issue*, *loan* and *transfer*.


Digital Item
Creator
Publisher
Distributor
Consumer
Down Stream Rights Specification: Grants, Licenses
Up Stream Rights Specification: Request, Royalties

Multi-tier Model

Interoperable

XrML has been designed to enable interoperability of components across DRM systems.

XrML provides syntactic interoperability, because it is encoded in XML and leverages system interoperability capabilities provided by XML technologies, such as XML parsers, that are compatible across platforms.

XrML provides semantic interoperability, because the specification provides normative definition of elements and element attributes, and language processing rules. Systems developed in compliance to the XrML specification will interpret the language the same way.

XrML provides system interoperability elements that are needed to interoperate on a system level. They include:

- Ensuring the authenticity of the XrML rights specification through the use and application of open and standard digital signatures technologies (W3C DSig). The language semantics enable protection of any semantically significant

[4] By design the XrML vocabulary in the current version is basic and thus certain terms such as "create" are not included. However, the extension mechanism allows adding these terms into the language.

[5] There are many definitions for Super-distribution. Here we refer a generic concept of to peer-to-peer distribution

construct in the rights expression via open and standard cryptographic mechanisms such as digital signature.

- Ensuring confidentially of the XrML document. Language semantics enable encryption of any semantically significant construct in the rights expression.
- Supporting web services through the use and specification of XML, XPATH, XSL SOAP, WSDL, and UDDI (all W3C submissions).
- Supporting pattern matching through XPATH (see pattern matching in the section on extensibility below).

Why specify Web Services?

Sometimes there is a need to communicate to a Web Service to handle rights-related operations, such as tracking state conditions (for instance, tracking the number of times a movie has been viewed) and performing revocation queries (determining whether the license still be trusted).

XrML enables this communication by including mechanisms for specifying the location of a particular service and the parameters to transmit.

WSDL describes the capabilities and functions that a web service can perform and its programmatic interface. SOAP is the protocol used to communicate with a web service, for example to make a request for service. UDDI is used by applications and businesses to discover these services in the Internet.

An XPath expression may be used to select data to be passed into the service. Examples of such expressions include a formula to select the issuer's details or to count the number of rights granted to the issued-to principal.

A style sheet may be included in the specification, which then is applied to the data selected by the XPath expression to transform them before passing them to the service.

Considerations on a trust model

A trust model is a central element of any system (such as a DRM system) that must determine whether a component can be trusted. When an application receives a rights specification, it must determine not only if it has been tampered with, but also if the issuing entity can be trusted[6].

One option in DRM is to require an end-to-end system design, which is bound by a common trust model outside of the language. This leads to a proprietary design with enforcement mechanisms described by the proprietary design and not by the rights language. Such is the case in all current DRM systems.

XrML adopts an explicit trust model approach that allows any number of trust domains to cooperate in the enforcement of rights. The trust specifications in XrML help the different parties determine to what extent, for what purpose, and when to trust other parties. As an end-to-end system design is no longer required, players are free to choose the best for each part of the system.

Extensible

XrML has defined a comprehensive set of rights, terms, and the semantics associated with them. The core elements of the language form a framework that enables

[6] A trust model requires security technologies such as encryption, digital signature, tamper detection, and so on

extensibility. We recognize that when the need arises, XrML must be capable of importing an additional set or sets of terms and adopting other standard ways of expression.

XrML is extensible via XML namespaces using XML schema technology. These extensions are defined in other namespaces.

Some extensions are designed specifically for XrML and substitute designated elements in the XrML core. This is called extensibility by substitution. For example, a new set of rights terms for a particular industry can be created which substitutes the current set for content. Different sets can be used to create more complex expressions.

Other extensions are designed independent of XrML but can be used by XrML. This is called extensibility through inclusion. For example, the metadata element defined in XrML has a limited number of attributes, but a metadata standard such as ONIX can be included as part of the expression.

Using Pattern Matching

XrML also uses pattern matching to extend how principals, resources, rights and conditions are specified. The general notion of a grant (an XrML construct) is to one principal for one right over one resource under one condition (however complex it might be). Pattern matching allows this general notion to be extended in a mathematical, unambiguous way to enable an issuer to make statements that involve an arbitrarily large and possibly infinite number of principals, rights, resources, or conditions.

For example, instead of generating a grant for a principal = john.doe@contentguard.com with another grant specifying principal = jane.doe@contentguard.com, pattern matching can be used to specify a grant with a principal = "anyone in the domain contentguard.com" if one wishes to apply the grant everyone.

Language SDK

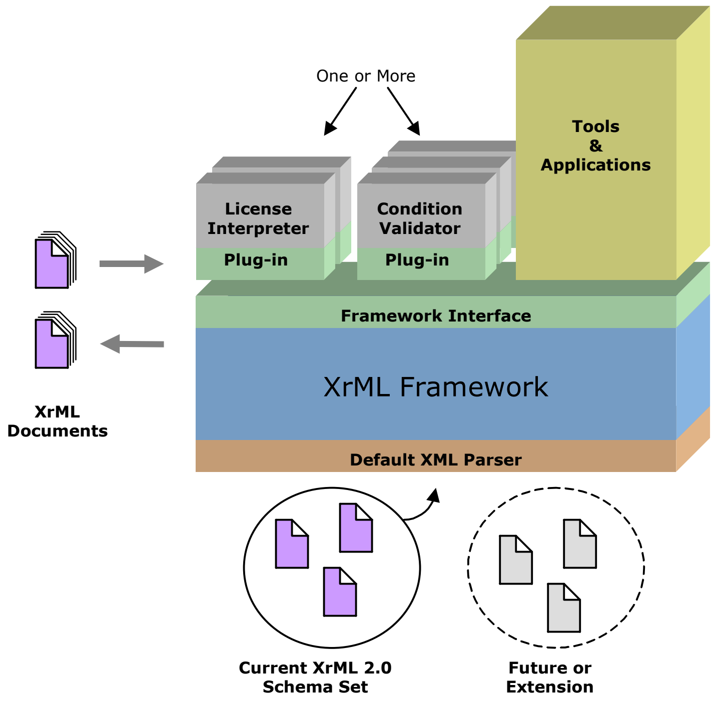
ContentGuard has leveraged its expertise in building end-to-end Digital Rights Management solutions and has developed a companion Software Development Kit (SDK) for XrML. Anyone developing a component or a system that deals with Digital Rights Management also must develop the components that process the rights expression. For example, a movie player that consumes a usage license and protected (encrypted) content must be capable of reading and understanding what the license allows the user to do. At the minimum, this involves reading, parsing, validating and interpreting the license.

The Language SDK consists of a Core Framework object and three components (Framework, Interpreter, and Condition Validator). The Core Framework is responsible for language manipulation functions such as loading a schema, reading the XrML file, creating a license, and invoking the appropriate interpreters. The Interpreter and Condition Validator are responsible for interpreting the rights and conditions specified in the XrML. These modules have been developed with an open interface, so that developers can build their own specialized interpreters and condition validators. The following diagram illustrates the XrML SDK architecture. For more information, please refer to the XrML SDK documentation.

By using the XrML Language SDK, developers and business can focus in their area of expertise and interface with XrML at a high level. For example, the movie player can query the license through a high level API, such as "can I play the movie now", instead of reading, parsing, validating and interpreting all the elements contained in the license.

The benefits of the Language SDK go beyond saving development time and costs. ContentGuard intends to continue its development in synch with the evolution of the language. A developer utilizing the XrML Language SDK will reap the following benefits:

- Backward and forward compatibility issues. These will be minimized and, in most cases, eliminated. An application using the SDK will rely on the SDK to deal with different versions of XrML.
- Signature and trust verification[7]. Signing and signature verification will be provided in the SDK, freeing developers from dealing with cryptographic mechanisms used to determine the integrity of the rights expression.
- Higher level of interoperability. The SDK has been developed by the same organization that developed the language. This means total adherence to the specification. Utilizing the SDK will reduce the risk that a rights expression may be interpreted differently in different systems.



XrML Language SDK

[7] This feature available post launch of the SDK

In short, the XrML Language SDK greatly enhances the business value of using XrML as the language to create and process rights expressions.

Conclusion

After a brief and highly inflated start, the DRM industry is currently in a consolidation mode. Mainstream industries are starting to deal with the issue of content and services that are associated with a license or rights. The shifting of desktop-centric applications to web services is an indication of this trend. In addition, increased awareness of digital distribution by the legal entities and the evolution of security technologies such as Public Key Infrastructure are encouraging the commerce of digital assets.

Digital Rights Management will play a central role in this emerging Internet landscape. Central to a DRM system is the mechanism to communicate rights information. XrML fulfills that need.

Additional Information

XrML Specification

XrML Cookbook