XrML 2.1 Technical Overview

DRAFT

Version 0.1

May 20, 2002

CONTENTGUARD™

Table of Contents

1.0 Introduction

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 specify and 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. These four elements are the Core of the language and determine the full context of the rights that are specified. Specifying the full context of the rights is important because it is not sufficient to just specify that the right to view certain content has been granted, but also *who* can view it and under *what* conditions.

Since its inception, the language has evolved through industry feedback, critical review, and product implementation. The language was adopted by the MPEG standards organization as the base of the MPEG rights language in 2001. Through this process, 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, 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 2.1.

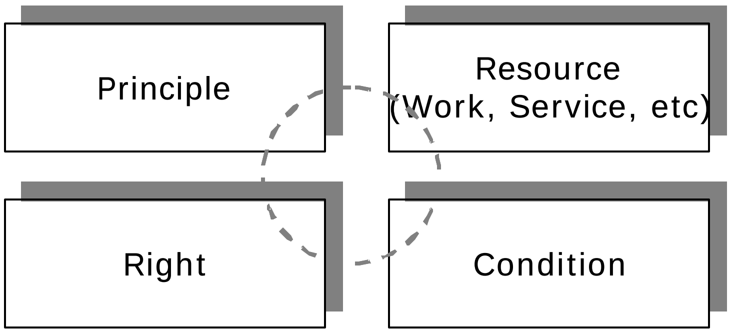
The purpose of this paper is to present the technical details of the language. A more general introduction to the language can be found in the *Need for a Rights Language* white paper and the definitive manual for the language can be found in the *XrML 2.1 Language Specification*.

2.0 XrML 2.1 Data model

XrML 2.1 adopts a simple and extensible data model for many of its key concepts and elements. The XrML data model consists of four entities and the relationship between those entities.

The basic relationship is defined by the XrML assertion "grant". Structurally, an XrML grant consists of the following:

- The principal to whom the grant is issued
- The right that the grant specifies
- The resource that is the direct object of the "right" verb
- The condition that must be met for the right to be exercised



2.1 Principal

A principal encapsulates the identification of a party to whom rights are granted. Each principal identifies exactly one party. In contrast, a set of principals, such as the universe of everyone, is not a principal.[1]

A principal denotes the party that it identifies by information unique to that party. Usefully, this information has some associated authentication mechanism by which the principal can prove its identity. The Principal type supports the following identification technologies:

- A keyHolder, meaning someone identified as possessing a secret key such as the private key of a public/private key pair. KeyHolders are represented using the KeyInfo technology from XML DSIG.
- A principal that must present multiple credentials, all of which must be simultaneously valid, to be authenticated.
- Other identification technologies that may be invented by others.

[1] Through the mechanisms of variable definition and pattern matching, XrML can define the principal as "any one in the universe". However, during the interpretation of the rights expression, the principal must be resolved to a single party--a specific entity or person in "the universe".

2.2 Right

A right is the "verb" that a principal can be granted to exercise against some resource under some condition. Typically, a right specifies an action (or activity) or a class of actions that a principal may perform on or using the associated resource.

The XrML 2.1 Core provides an abstract right element to encapsulate information about rights. It also provides a set of commonly used, specific rights relating to other rights, such as issue, revoke, and obtain. Extensions to the XrML Core define rights that appropriate to using specific types of resources. For instance, the XrML Content Extension[2] defines rights appropriate to using digital works (for instance, play and print rights).

2.3 Resource

A resource is the "object" to which a principal can be granted a right. A resource can be a digital work (such as an e-book, an audio or video file, or an image), a service (such as an email service, or B2B transaction service), or even a piece of information that can be owned by a principal (such as a name or an email address).

The XrML 2.1 Core provides mechanisms to encapsulate the information necessary to identify and assign rights to a particular resource. Patterns allow identification of a collection of resources with some common characteristics. Extensions to the XrML Core define resources appropriate to specific business models.

2.4 Condition

A condition specifies the terms, conditions, and obligations under which rights can be exercised. A simple condition is a time interval within which a right can be exercised. A slightly more complicated condition is to require the existence of a valid, prerequisite right that has been issued by some trusted entity. Using this mechanism, the eligibility to exercise one right can become dependent on the eligibility to exercise other rights. Moreover, a list of conditions can be put in conjunction to form a condition requiring that the conditions all be met simultaneously.

The XrML 2.1 Core defines an abstract condition element to encapsulate information about conditions and some very basic conditions. Extensions to the XrML Core could define conditions appropriate to specific distribution models. For instance, the XrML Content Extension (currently being developed by a separate standards organization/activity) defines conditions appropriate to using digital works (for instance, watermark, destination, and renderer).

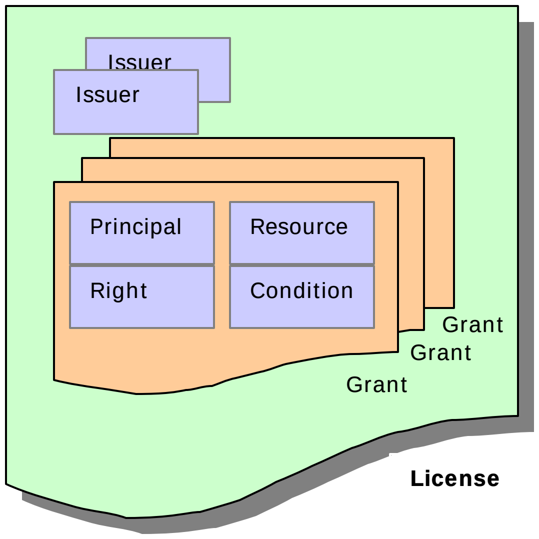
[2] The XrML Content Extension was part of the XrML 2.0 specification, In XrML 2.1, the Content Extension is no longer part of the specification, but a separate standards activity (in this case the MPEG organization) will be responsible for developing the Content Extension for XrML

2.5 Data Model is Encapsulated in XML Core Schema

Since XrML is defined using the XML Schema recommendation from W3C, its element model follows the standard one that relates its elements to other classes of elements. For example, the “grant” element is related to its child elements, “principal”, “right”, “resource”, and “condition”.

3.0 XrML Basic Data Constructs

As indicated in the previous section, the basic XrML construct is the assertion “grant”. In a system environment, there is the additional need to determine other things such as the identification of the principal who issued the grant and the identification of the license. For this reason, in a system environment, the central XrML construct is a “license”. Conceptually, a license is the issuance of grants by their issuing parties.



3.1 License

The basic structure of a license contains the following:

- A set of grants that convey to certain principals certain rights to certain resources under certain conditions
- An identification of the principal or principals who issued the license and thus bestow the grants upon their recipients
- Additional information such as a description of the license and validity date

Those familiar with digital certificates and other similar structures might notice the absence of the identification of the principal or principals to whom certain rights are conveyed. This notion has been regularized within the structure and terms of each individual grant.

The license issuer

The principal who issues a license can digitally sign it, signifying that the issuer does indeed bestow the grants contained in it. Syntactically, multiple issuers may sign a given license. However, no additional semantic is associated with the joint signing; it is as if each had signed a copy of the license independently.

3.2 Grant

A grant is the element within the license that bestows an authorization upon some principal. It conveys to a particular principal the sanction to exercise an identified right against an identified resource, possibly subject to first fulfilling some conditions.

Structurally, a grant consists of the following:

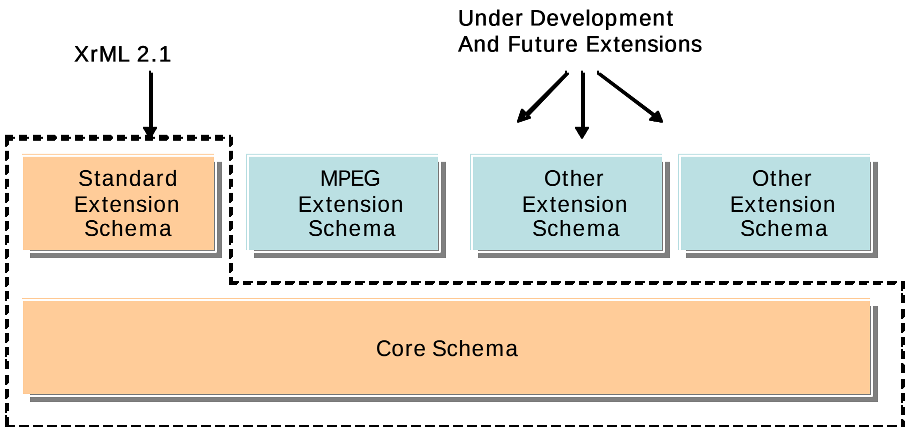
- The principal to whom the grant is issued
- The right that the grant conveys to the specified principal
- The resource against which the specified principal can exercise or carry out this right
- The condition that must be met before the right can be exercised

4.0 Structure and Organization of the Language

The syntax of XrML 2.1 is described and defined using the XML Schema technology defined by the Worldwide Web Consortium (W3C). Significantly more powerful and expressive than DTD technology, the extensive use of XML Schema enables XrML 2.1 to offer a high degree of richness and flexibility in its expressiveness and extensibility.

To that end, a principal design goal for XrML 2.1 is to enable and support a significant amount of extensibility and customizability without requiring actual changes to the XrML 2.1 Core. Indeed, XrML 2.1 makes use of this extensibility internally, even within the Core itself.

XrML 2.1 is organized into several parts:



- A Core Schema containing definitions of concepts that are at the heart of the XrML 2.1 semantics, particularly those having to do with evaluation of a trust decision.
- A Standard Extension schema containing definitions of concepts that are generally and broadly useful and applicable to XrML 2.1 usage scenarios, but which aren't necessarily at the heart of XrML 2.1 semantics

XrML 2.1 is typically used in conjunction with one or more extensions. For example an extension related to digital content such as the one under development by the MPEG group.

Each of the two XML Schemas is a normative part of the overall XrML 2.1 specification. In particular, the Core Schema is a normative part of the XrML 2.1 Core. Other parties may, if they wish, define their own extensions to XrML 2.1. This is accomplished using existing, standard XML Schema and XML Namespace mechanisms.

4.1 Mandatory vs. Optional terms

Most of the terms defined by the language are, in fact, optional. Only certain elements are mandatory in order to produce a valid XrML license conveying at least one grant[3]. The schema clearly indicates this property. The use of mandatory and optional terms allows the creation of both simple and complex expressions. Optional terms are used as needed.

The set of mandatory elements is organized as follows:

- ***License***
- ***Grant*** or ***GrantGroup***
- ***Right (abstract)*** or one of its substitutions

Right is abstract, and it is replaced by Issue, Obtain, PossessProperty, Revoke or any right defined in an extension (for example the Play right defined in the Content Extension).

All the mandatory terms for defining a License are part of the Core Schema. All the terms in the Standard and any other extension schema such as a content extensions are optional.

The notion of the mandatory set within the Core is that of a ***license granting a right***. When the right used is one of those in the Core, then the notion is that of establishing an identity and/or establishing certain rights to manipulate rights

PossessProperty is used to establish the digital identity of a principal. Precisely identifying a principal is the key element for the management of digital rights.

Rights (and conditions) are industry specific, for example a digital movie is played but cannot be printed. An e-book can be viewed and printed. Resources are also industry specific: a digital movie, a picture, a text manuscript, a file system, etc.

[3] If one is not constructing an ***XrML License***, then there are no mandatory terms. In other words, one can create a valid XrML expression with any terms defined in the language.

Example of a minimal license

A simple and minimal license can be constructed using the mandatory terms and a couple of optional terms. The following license certifies that the holder of the (cryptographic) key has the common name “Alice Richardson”

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Copyright (C) 2001 ContentGuard Holdings, Inc. All rights reserved.
   "ContentGuard" is a registered trademark and "XrML", "eXtensible rights Markup
   Language", the XrML logo and the ContentGuard logo are trademarks of
   ContentGuard Holdings, Inc. All other trademarks are properties of their
   respective owners.-->
<!-- This is a simple certificate -->
<license xmlns="http://www.xrml.org/schema/2001/11/xrml2core"
     xmlns:sx="http://www.xrml.org/schema/2001/11/xrml2sx"
     xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.xrml.org/schema/2001/11/xrml2cx..\schemas\xr
     ml2cx.xsd">
   <!-- Certify that the following key holder has the common name "Alice
     Richardson"-->
   <grant>
     <keyHolder>
       <info>
         <dsig:KeyValue>
           <dsig:RSAKeyValue>

                  <dsig:Modulus>Fa7wo6NYfmvGqy4ACSWcNmuQfbejSZx
                  7aCibIgkYswUeTCrmS0h27GJrA15SS7TYZzSfaS0xR9lZ
                  dUEF0ThO4w==</dsig:Modulus>
               <dsig:Exponent>AQABAA==</dsig:Exponent>
             </dsig:RSAKeyValue>
           </dsig:KeyValue>
         </info>
       </keyHolder>
       <possessProperty />
       <sx:commonName>Alice Richardson</sx:commonName>
     </grant>
   </license>
```

As indicated above, the starting point or the basic premise for the management of Digital Rights is establishing an identity. The expression of most rights depends on the type of industry. Note that the above example has no issuer. Although, an issuer may be required in most applications, the example is a valid XrML license.

5.0 System Features of the Language

One of the advanced features of XrML is the comprehension of system issues for interoperability. In other words, in a real system, specification of rights is not sufficient for interoperability. Some of the additional issues that must be addressed are:

- Can the systems trust the licenses?
- Can the language be extended without breaking the core specification?
- Can the systems interact with services that are part of, but outside of, a system such as a DRM system?

XrML has been designed to address these issues and to enable interoperability of components across systems.

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

- Trust specification: 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 construct in the rights expression via open and standard cryptographic mechanisms such as digital signature.
- Confidentiality: Ensuring confidentially of the XrML document by enabling encryption of any semantically significant construct in the rights expression.
- Web Service Specification: Supported through WSDL and UDDI.
- Pattern Matching: Supported through XPATH.

5.1 Trust Specification

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[4].

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

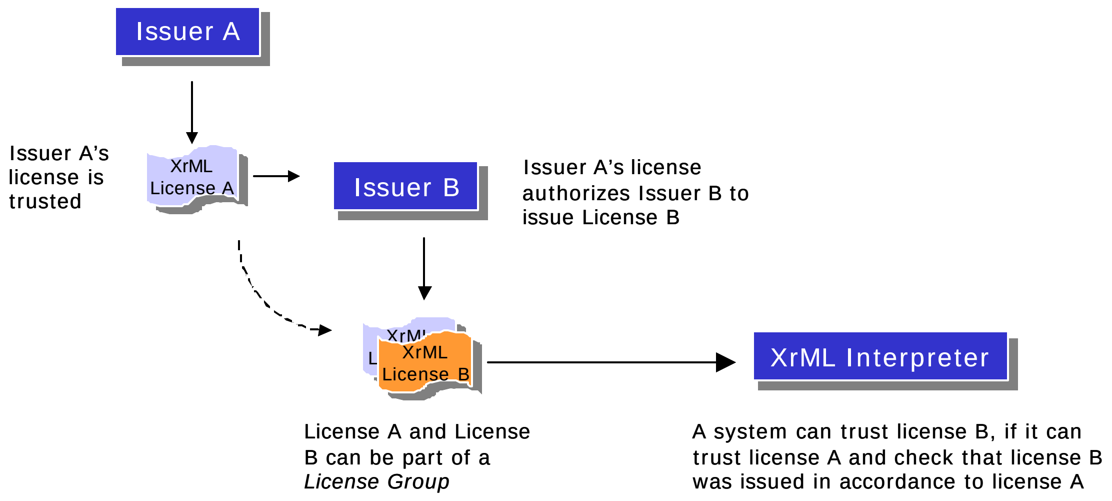
[4] A trust model requires security technologies such as encryption, digital signature, tamper detection, and so on

trust other parties. As an end-to-end system design is no longer required, players are free to choose the best design for each part of the system.

XrML defines "issuer" and "trustedPrincipal", and allows the definition of licenses that specify the right to issue other licenses to encapsulate the explicit trust model. The issuer element in a License contains two pieces of information: a set of issuer-specific details about the circumstances under which he issues the License (dates, revocation mechanisms, etc), and a digital signature for the License. With this, the decision point is whether or not a system would trust and validate that signature. TrustedPrincipal indicates a policy by which Principals are identified as having the appropriate and necessary qualifications in order to be trusted for use in certain situations.

In many cases, a system may blindly trust that the issuer's signature is his own (unless it is known that the key has been hacked). However, a system should not blindly trust the issuer's right to issue whatever license has been issued. A system will usually have some sort of check, either:

- The system trusts the issuer itself (and thereby trust everything the issuer signs). This model is typical of a closed-trust system (monolithic trust). All the components work under a single certificate authority and all the signatures are trusted (unless revoked).
- The system sees a license that gives the issuer (issuer #1) the right to issue and the system trusts the second issuer's (issuer #2) right to issue that right to the first issuer (issuer #1). This is what XrML enables a system to check. The check for trust can chain up to a signature (or license) that is trusted by default. In this scenario, an "open-trust" environment is possible, where the trust of a license can be determined (or validated) by checking the licenses that have been used to issue subsequent licenses (see the figure below)
- The system trusts the issuer's right to issue the license by relying on some out-of-band means. For instance, the system knows that "Alice" is the author of a book and so it trusts "Alice" to issue people the right to play the book.


Issuer A
Issuer A's license is trusted
XrML License A
Issuer B
Issuer A's license authorizes Issuer B to issue License B
XrML License B
XrML Interpreter
License A and License B can be part of a *License Group*
A system can trust license B, if it can trust license A and check that license B was issued in accordance to license A

5.2 Confidentiality

Certain business models require that the details about rights and conditions be kept confidential. XrML defines EncryptedLicense and EncryptedGrant for this purpose. EncryptedLicense provides a mechanism by which the contents of a License may be encrypted and so hidden from view from inappropriate parties. EncryptedGrant indicates that the grant is encrypted and so hidden from view from inappropriate parties. This mechanism makes straightforward use of the XML Encryption Syntax and Processing standard and enables systems to understand how each part of a license has been protected through encryption.

5.3 Web Service Specification

Web service specification leverages the serviceReference element in XrML. a serviceReference indicates the location and the means and manner by which a client is to interact with a specific service. Specifically, a serviceReference element does the following:

- Identifies the location or address at which the service is found.
- Identifies a greater or lesser amount of metadata about the semantics of the service and the rules to which the client must adhere to interact with it
- Optionally specifies a set of concrete parameters that are to be provided when a client interacts with the service by dereferencing this particular serviceReference. These parameters provide a means by which a service might at run time distinguish between its uses from different XrML 2.1 contexts.

XrML 2.1 does not itself invent significant new infrastructure for describing services; rather, it draws on the considerable work being done in this area by others. Specifically, there are two architected technologies by which the location and metadata information of a ServiceReference may be provided (using an xsd:any element, ServiceReference provides for other technologies that may also be used):

- the Web Services Definition Language (WSDL)
- the Universal Description, Discovery, and Integration (UDDI) directory infrastructure

5.4 Pattern Matching

XrML uses pattern matching to specify sets of principals, rights, resources and conditions. 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). It is then quite useful and important at times to be able to write in XML formal expressions that semantically denote particular sets of resources or other elements.

XrML defines the following elements to support pattern matching:

XmlPatternAbstract: All formal patterns in XrML 2.1 have types that derive from the type XmlPatternAbstract.

XmlExpression: XmlExpression provides a means by which patterns written in formal expression languages defined outside of XrML 2.1 can be straightforwardly incorporated. The particular expression language used is indicated by the **lang** attribute, which is a URI. The default value for **lang** is http://www.w3.org/TR/1999/REC-xpath-19991116, which indicates that the contents of the XmlExpression contains a string which is an XPath expression.

PrincipalPatternAbstract, RightPatternAbstract, ResourcePatternAbstract, and ConditionPatternAbstract: As an alternative to using patterns written in externally-defined expression languages, it is often useful to define new XML types and elements that, in their intrinsic semantic, define some pattern matching algorithm. This can, of course, be done by simply deriving from XmlPatternAbstract, but, in some situations, deriving from one of these four types might be more useful.

6.0 Summary

XrML has been carefully designed to provide a concise yet comprehensive set of elements and a grammar to construct rich expressions for rights. Its simple, yet flexible, data model is practical for both simple and complex expressions for diverse business models: from a simple specification of rights for a user viewing some content, to complex models for multi-tier distribution of content. We believe that XrML has achieved the "right balance" between compactness and comprehensiveness in an XML based language.

Because of its capability to address many different models, XrML is well positioned to be the language for interoperability between components and systems that deal with rights and conditions (such as DRM systems, Content Management Systems, Licensing Mechanisms, and so on).

By using contemporary standards, the language is extensible, open, and ready to meet the interoperability requirements for web services.

In short, XrML has been developed to be obsolesce-proof; simple, yet highly expressive; and forward looking to business models in the future.