XrML: Extensible rights Markup Language

http://www.xrml.org

PLEASE NOTE THAT YOU HAVE TO ACCEPT THE TERMS AND CONDITIONS SET FORTH IN THE XRML LICENSE AGREEMENT BEFORE YOU USE THIS DOCUMENT AND THE XRML DTD. IF YOU RECEIVED THIS DOCUMENT AND HAVE NOT SIGNED THE LICENSE AGREEMENT, PLEASE GO TO XrML.org TO SIGN THE LICENSE AND GET THE XRML SPECIFICATIONS AND DTD. IF YOU USE THIS DOCUMENT AND HAVE NOT YET SIGNED THE LICENSE, PLEASE READ THE FOLLOWING CAREFULLY, AS ANY USE YOU MAKE OF THE XRML SPECIFICATIONS AND DTD WILL BE SUBJECT TO THE FOLLOWING TERMS AND CONDITIONS. BY USING THE XRML SPECIFICATIONS AND DTD, YOU INDICATE THAT YOU AGREE TO THESE TERMS.

THE FOLLOWING IS THE LICENSE AGREEMENT RELATING TO THE XrML SPECIFICATIONS YOU REQUESTED. YOU SHOULD TAKE SOME TIME TO READ THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT CAREFULLY. BY SELECTING „ACCEPT“ YOU ARE AGREEING TO ALL OF THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT. IF YOU ARE AGREEING ON BEHALF OF A COMPANY, YOU REPRESENT THAT YOU ARE AUTHORIZED TO BIND THAT COMPANY TO THIS LICENSE AGREEMENT. IF YOU DO NOT AGREE TO THESE TERMS AND CONDITIONS, SELECT THE „REJECT“ BUTTON AT THE END. IF YOU ACCEPT THIS LICENSE, CONTENTGUARD WILL SEND YOU THE XrML SPECIFICATIONS ELECTRONICALLY.

PREAMBLE:

ContentGuard, Inc. is committed to promoting and supporting a standard language that will enable content creators, providers, distributors and retailers to express rights and specifications that can be universally interpreted by trusted systems technology platforms. To facilitate this common standard ContentGuard has developed the XrML Specifications for digital rights management software (based on extensive research conducted at the Palo Alto Research Center - PARC) and desires to license the use of the XrML Specifications royalty-free.

GUIDING PRINCIPLES FOR XrML:

1. Enable rapid growth of the eContent industry for all media.
2. Enable XrML to meet the needs of all stakeholders in the eContent industry.
3. Establish a community of practice that is committed to develop a common rights language for use by trusted systems.
4. Enable interoperability across multiple platforms and content types.
5. Encourage interested parties to submit and share XrML Modifications with the community of practice that will extend and enhance the XrML Specifications.

6. Enable future XrML enhancements and modifications to meet the needs of the DRM community, and that the standard does not become fragmented or create commercial advantage for any single party.
7. Establish a XrML Panel of knowledgeable and interested parties, to review the XrML Modifications and make recommendations concerning incorporation of XrML Modifications in the XrML Specifications.
8. This License Agreement does not apply to, dictate or restrict General-purpose APIs (Application Program Interfaces), operating system functions and specifications, or other software that incorporate XrML Specifications.
9. Additional materials will be made available on the xrml.org website from time to time, for example, the DTDs for the XrML Specifications.

A. DEFINITIONS:

1) "Digital Rights Management" or "DRM" means techniques, processes, procedures and algorithms related to establishing an environment that utilizes syntactically expressed declarative statements having an environment-wide meaning for the management of digital rights, including computer hardware and software, which enables or implements trusted licensing, secure rights and permissions specification, rights and permissions enforcement, establishment of a trusted computing environment, and trusted infrastructure, each for:
   (i) the secure preparation, transmission, prevention of misuse and/or consumption of protected digital works by authorized licensees, (such as watermarking and fingerprinting and software obfuscation;) and
   (ii) secure digital commerce transactions, including the automated and persistent enforcement of policies for consumption of digital goods, usage tracking, budget management, fraud detection, account management, key management and similar functions, to occur over internal and external computer networks.

2) "XrML" or "eXtensible rights Markup Language" means the rights specification language for implementing DRM owned by ContentGuard.

3) "XrML Specifications" means the specifications and documentation for implementing XrML as first provided to You under this License Agreement and any future versions, enhancement, improvements and updates thereof that may be provided, or otherwise made available to You by ContentGuard under this License Agreement.

4) "Licensee" means any party (including You) that has entered into a license agreement with ContentGuard for the license of the XrML Specifications.

5) "XrML Modifications" means any modifications, enhancements, improvements, changes, and Derivative Works based on the XrML Specifications.

6) "Derivative Work" has the meaning set forth therefore in Title 17 United States Code Section 101.

7) "You" means an individual, or a legal entity acting by and through an individual or individuals, who receives the XrML Specifications under this License agreement or who exercises rights under this License Agreement or who uses the XrML

Specifications for any purpose. For legal entities, "You" includes any entity that by majority voting interest controls, is controlled by, or is under common control by You.

8) "XrML Panel" will consist of knowledgeable and interested representatives of Licensees interested in creating a common DRM standard.

B. LICENSE GRANTS AND OBLIGATIONS:

1. ContentGuard License Grant to You:

Subject to the terms and conditions of this License Agreement, ContentGuard hereby grants to You a non-exclusive, world-wide, royalty-free license to use the XrML Specifications for:

(i) any application, product or service incorporating, providing or using Digital Rights Management functionality; and
(ii) research and training use.

2. XrML Modifications:

(i) XrML Modifications. By entering into this License Agreement and by using the XrML Specifications, You agree that You share the goal of a common standard for DRM supported by the XrML Specifications. To achieve this goal, ContentGuard may review the XrML Specifications from time to time to ensure that the XrML Specifications continue to define and support a common standard for DRM. As a result of this review, ContentGuard may issue future versions, enhancements, improvements, and updates to the XrML Specifications. ContentGuard may also incorporate XrML Modifications made by ContentGuard or third parties into the XrML Specifications.

(ii) Process for Incorporating XrML Modifications into the XrML Specifications. ContentGuard encourages and welcomes You to offer XrML Modifications for enhancing and extending XrML Specifications. By entering into this License Agreement You agree that only ContentGuard, in its sole discretion, may incorporate the XrML Modifications made by You or any third party into the XrML Specifications. ContentGuard and the XrML Panel will review all XrML Modifications. If Your XrML Modification is not adopted by ContentGuard, ContentGuard will make reasonable endeavors to explain to You why Your XrML Modification was at variance with the XrML Specifications standard.

(iii) XrML Modifications. Subject to the terms and conditions of this License Agreement, ContentGuard hereby grants to You the right to make XrML Modifications, provided that:
  (a) You promptly provide ContentGuard with copies of all XrML Modifications made by or on behalf of You;
  (b) for any XrML Modification made by You which is incorporated by ContentGuard in the XrML Specifications, You will grant (and You agree this License Agreement constitutes such grant), to ContentGuard a

perpetual, world-wide, royalty-free, irrevocable, non-exclusive license, including the right to sublicense, in the XrML Modification (and agree that this License Agreement shall constitute such assignment); and

(c) If ContentGuard informs You in writing that any XrML Modification made by You is not approved for integration into the XrML Specifications, You may use such XrML Modification but You must not designate, advertise or promote in any way that any such XrML Modification is compatible with the XrML Specifications and You agree that You will cease using the notation described in Section B5(ii) of this License Agreement.

3. Use of the XrML Specifications for DRM Functionality:

You agree that You will use all reasonable efforts to ensure that any applications, products or services developed by You will not be incompatible with the approved XrML Specifications. If Your applications, products or services use XrML Specifications then it must support XrML as well as that of any other rights specification.

4. Future Versions of XrML Specifications:

From time to time ContentGuard may issue or otherwise make available to You, such as by download at the ContentGuard XrML.org website, new versions, improvements and updates to the XrML Specifications.

5. Your Obligations:

Subject to the terms and conditions of this License Agreement, You agree to:

(i) note in conspicuous locations on any of Your products and services (and related marketing materials and user documentation) that use the XrML Specifications or any XrML Modifications made by You the notation: "XrML, eXtensible Rights Markup Language and the XrML logo are trademarks of ContentGuard Holdings, Inc. Copyright © 2000 ContentGuard Holding, Inc. All rights reserved;"

(ii) Insert and maintain in conspicuous locations on any of Your products and services (and related marketing materials and user documentation) that use the XrML Specifications or any XrML Modifications made by You the notation: "XrML Compatible" and the XrML logo. The logo (which ContentGuard reserves the right to modify from time to time) is available at the XrML web page www.xrml.org.

(iii) not sue ContentGuard or any other Licensee for claims arising out of the use of XrML Specifications or XrML Modifications made by You;

(iv) not sublicense, or disclose the XrML Specifications made by You to third parties except where this is part of the distribution of Your products or services using XrML Specifications; and

(v) not to sue ContentGuard, Licensees and other third parties for using the XrML Specifications and XrML Modifications.

C. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS LICENSE, THE XrML SPECIFICATIONS ARE PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

D. DISCLAIMER OF LIABILITY

CONTENTGUARD SHALL NOT BE LIABLE FOR ANY DAMAGES, DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, PUNITIVE, CONSEQUENTIAL OR OTHER DAMAGES (INCLUDING, WITHOUT LIMITATION, LOST PROFITS BUSINESS INTERRUPTION, LOSS OF PROGRAMS OR OTHER INFORMATION OR OTHER LOSS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THIS LICENSE AGREEMENT, THE USE OR DISTRIBUTION OF THE XRML SPECIFICATIONS OR XRML MODIFICATIONS OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. CONTENTGUARD SHALL NOT BE LIABLE FOR ANY CLAIM PERTAINING TO ERRORS, OMISSIONS, OR OTHER INACCURACIES IN THE XRML SPECIFICATIONS OR ANY XRML MODIFICATIONS. IN THIS PARAGRAPH "CONTENTGUARD" INCLUDES ITS SUBSIDIARIES, PARENTS, EMPLOYEES, OFFICERS, DIRECTORS, AGENTS, SUBCONTRACTORS, SERVICE PROVIDERS AND SUPPLIERS.

E. GENERAL PROVISIONS

1. Integration: This License Agreement represents the complete agreement concerning the subject matter hereof.
2. Governing Law: The laws of the State of New York and the intellectual property laws of the United States shall govern this License Agreement. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded.
3. Severability: If any provision of this License Agreement is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable.
4. Termination: Your rights under this License Agreement shall terminate if You fail to comply in a material respect with the terms of conditions of this License Agreement and do not cure such failure within 60 days after receiving written notice of such noncompliance from ContentGuard. You have the right to terminate this License Agreement effective 60 days after written notice to ContentGuard.
5. Language: This License Agreement is in the English language only, which language shall be controlling in all respects, and all versions of this License Agreement in any other language shall be for accommodation only and shall not be binding on the parties to this License Agreement.
6. Export Laws: You agree to comply strictly to with all applicable import and export laws and regulations of the United States and other countries.

Abstract

This specification defines XrML (eXtensible rights Markup Language), an extension of Xerox DPRL 2.0 in XML. XrML documents provide descriptions of usage rights, fees and conditions for digital contents, together with message integrity and entity authentication in these descriptions.

Status of this document

Although the extensible rights markup language (XrML) has already benefited from comments from publishers and other interested parties, the language is not frozen. Currently, this document is a draft. Its description of works and rights is mainly from DPRL v.2. Rules for generating and processing XrML documents are still in discussion. Feedback and suggestions are solicited.

Public discussion on XrML features takes place on the discussion forum, email: xrml@xrml.org, WWW http://www.xrml.org/forum.stm

Please report errors in this document to the current editor at editor@xrml.org.

Revision History

Date	Change Description	Edited By
4/28/2000	Initial draft of version 1.0	X. Wang
5/10/2000	Incorporated modification of the DTD: – Added element AUTHENTICATEDDATA to element BODY – Removed element PRINCIPAL from element ENABLINGBITS – Added element PRINCIPAL to element PRINCIPAL	X. Wang
6/23/2000	– Some typo fixing – Added element SECURITYLEVEL to element PRINCIPAL	X. Wang

1 INTRODUCTION

1.1 What is XrML?

XrML is the shorthand for eXtensible rights Markup Language. It is a language in XML (eXtensible Markup Language) for describing specifications of rights, fees and conditions for using digital contents (or properties), together with message integrity and entity authentication within these specifications. It is intended to support commerce in digital contents, that is, publishing and selling electronic books, digital movies, digital music, interactive games, computer software and other creations distributed in digital form. It is also intended to support specification of access and use controls for secure digital documents in cases where financial exchange is not part of the terms of use. XrML documents are XML conforming. As such, they are readily viewed, edited, and validated with standard XML tools.

The use of XrML for usage rights on digital contents is to ensure that trusted systems can exchange digital contents and interoperate. Trusted systems (or repositories) are systems that can hold digital contents and which can be trusted to honor the rights, conditions, and fees specified for digital contents. In document commerce, trusted systems are for authoring, playing, and selling digital works. They include personal systems, on-line storefront systems, library systems, and others.
Design goals for XrML are:

- To enable content owners and distributors to describe rights, fees and conditions appropriate to commerce models they select.
- To provide standard terms for usage rights with useful, concise, easily understandable meanings.
- To offer vendors sound operational definitions of trusted systems for compliance testing and evaluation.
- To provide extensibility to new language features to meet the needs of the digital content industry today and as it develops in the future.
-

XrML 1.0 (this specification) is an extension of the Xerox "Digital Property Rights Language" [DPRL] version 2. DPRL, as originally conceived, was to be a language for describing digital properties (works) and their usage rights, fees and conditions. Its author is Mark Stefik from Xerox PARC. Its latest version (version 2) is in XML. DPRL has been around since 1996 and has generated several U.S patents around it. In order to build trusted systems based on documents specified in DPRL, integrity of these DPRL documents and authentication of entities defined and/or identified within the documents become important. Moreover, several important usage conditions in digital rights management, such as usage location and tracking were not defined in DPRL. XrML addresses these issues and extends DPRL by adding a set of structural and semantic tags suitable for specifying metadata of XrML documents, validating integrity of XrML documents as well as of digital contents, and authoring relatively simple XrML documents (such as licenses). In addition, XrML adds support for specifying conditions for usage locations and tracking, and simplifies some DPRL document elements and their attributes.

1.2 Basics of Digital Property Rights

Digital property rights (or “usage rights” for short) are rights associated with digital works and their parts that describe how the works can be used. Here are some basic concepts:

- Rights are associated with parts of a digital work (and with folders).
- Every class of usage right has a corresponding transaction.
- The transaction defines what a repository does when the right is exercised.
- Rights are described in sentences of a machine-interpreted language having a grammar.
- The transactions for a given work are parameterized by the information in the usage rights sentences for the work.

- The rights on a work can be changed later, if the change is authorized by the rights owner.

1.3 Scope of this Document

This document explains the basic concepts for managing digital works in trusted systems, describes the language syntax and semantics, and provides examples of typical specifications of usage rights. It does not provide specifications for security in trusted systems, propose specific applications, or describe the details of the accounting systems required.

One of the goals of this document is to develop an approach and language that can be used throughout the publishing industries and in other industries as well. This document does not address the agreements, coordination or institutional challenges involved in achieving that goal.

1.4 Interfaces to Digital Property Rights Specifications

Specifications in the rights markup language are established by creators of digital works and also by publishers and distributors. Specifications are used by consumers as they select and use digital works. However, creators, publishers, distributors, or consumers would seldom (if ever) look directly at specifications written in the language.

Thus, specifications in the rights markup language are intended to be readable by machines rather than directly readable by people. They are used by systems that store, play, transport, copy, and sell digital works. These repositories potentially include a range of devices with embedded computers, including personal computers, set top boxes, personal document readers, personal entertainment devices, on-line servers, and trusted printers.

In many cases, the rights associated changing rights with a digital work are established at the time that the work is published and remain the same over time. There are, however, several ways that rights may be changed over time, and these involve different interfaces.

- *Republishing with new rights*. In the simplest case, the publisher or rights holder may simply republish a work with new rights. At the time of publishing, the terms and conditions on new copies of the work are changed to be different from those on older copies of the work. One way to accomplish this is simply to reinvoke a rights editor on a copy of the work and then to begin distributing only the revised version. An important job of a rights editor is to check the credentials of the person seeking to change the rights, since only the rights owner is authorized to change the rights on a work.
- *Embedding a work and adding new terms and fees*. In this case, a work may be embedded in a larger work. Embedding is controlled by the Embed right discussed later in this document. Generally, embedding is done by a distributor, who adds value by repackaging the work. Embedding typically adds extra fees for exercising rights. Although this operation typically includes combining a work with additional content, it includes the case where the content itself is not changed.
- *Upgrading rights*. Some trusted systems will support the operation of changing the rights on a work already distributed. This process would be carried out on a trusted system which checks the authorization for the change during a transaction that changes the rights. The details of the transaction are beyond the scope of this document except to say that the operation is automatic and the transaction to change the keys must be digitally signed by the rights owner.

Programmers developing trusted applications that use the rights language would typically not work with the syntax directly. Instead, they would use the application programmer's interface (API) to an object-oriented representation of the statements in the rights language. The application interface is described in other documents.

1.4.1 Rights Editor Interfaces

To specify rights a publisher of a digital work uses a graphical user interface (a “rights editor” program) in a digital authoring or publishing system. Such interfaces can take different forms. For example, one convenient form of a right editor interface is similar to a spreadsheet.

The interface enables a publisher to fill in information indicating what rights, fees, or licenses apply to the digital work at hand. For example, a publisher may simply select individual rights or a set of standard rights from a menu, modifying them slightly to indicate particular fees or distribution requirements. The publishing program then outputs a specification written in the rights markup language and attaches that specification to the digital work.

1.4.2 Trusted System Interfaces

Trusted systems, such as trusted workstations, trusted music players, trusted video players, and other trusted devices need user interfaces appropriate to the device.

For example, to play a work of digital music or a video, a consumer would just press the play button of a trusted player in much the same way that one uses a play button on a conventional compact disk or video tape player. In many systems, pushing a button would automatically invoke all of the appropriate checking, authorization, and payment transactions that are required for playing the digital work. In some systems, a consumer could use a small display similar to a calculator display to select among different rights or billing options. Different devices would have different user interfaces, but interoperable devices would share a common rights markup language — a *lingua franca* for systems dealing in permissions and fees.

In some cases, a digital work may be offered with hundreds of variations on the rights. For example, a publisher may offer variations on rights at different prices for different users. There may be regular consumer rates, corporate rates, special rates for academic institutions or libraries, special rates for members of the book of the month club, special rates for frequent fliers, special rates for recent purchasers of other works by the same publisher, and so on. There can be different rates for a permanent use, use over a ten year period, or use over a special two-day introductory period.

In many cases, an unmediated interface to so many variations will be overly complex for normal use. An alternative is for the system to present the set of credentials, memberships, and special offers available to the user via an interface agent that matches held certificates and tickets with the rights offers available on the digital work. Given a selection policy established by the user, the agent can filter the choices down to a small set or automatically select the lowest cost right meeting the user’s needs. For example, on a trusted music player, pressing the play button may be all that is required to select the least expensive alternative and play the work.

1.5 Editorial Conventions

For readability, brevity, and historic reasons this document uses the words digital property rights, digital rights, and usage rights synonymously. Similarly, this document uses the words extensible rights markup language, rights markup language, digital property rights language, digital property language, digital rights language, and rights language synonymously. It also uses the terms digital publications, digital documents, digital property and digital works synonymously.

This specification uses XML DTDs as a main tool to define the XrML grammar. (Readers unfamiliar with DTD syntax may wish to refer to Ron Bourret's "Declaring Elements and Attributes in an XML DTD" [Bourret].)

This document uses capitalized keywords to specify XrML elements and lower-case keywords to define attribute names. For instance, a right to print is specified as <PRINT maxcount ="5"> … </PRINT>.
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in RFC2119.

This specification uses XML-namespaces. It uses Uniform Resource Identifiers [URI] to identify resources, algorithms, and semantics. The URI in the namespace declaration above is also used as a prefix for URIs under the control of this specification. For resources not under the control of this specification, the designated Uniform Resource Locators [URL] are used.

2 XrML OVERVIEW AND EXAMPLES

This section provides an overview and examples of XrML syntax. The formal syntax is found in Section 3: Top-Level Syntax and section 4: Work and Rights Syntax. The specific rules for generating and processing XrML documents are given in section 5: Generating and Processing Rules.

In this section, an informal representation and examples are used to describe the structure of the XrML syntax. This representation and examples may omit attributes, details and potential features that are fully explained later.

2.1 XML notions

XrML documents are specified using the element/attribute markup model of eXtensible MarkUp Language (XML) .

Terms in CAPITAL represent elements. An element may contain other elements, together with character data. Elements can also contain meta-data in the form of attribute list. A basic XML element declaration looks like this:

```
<!ELEMENT LIST (ITEM)+>
```

This means that the element `LIST` contains one or more `ITEM` elements. "+" denotes one or more. "*" denotes zero or more. A "?" indicates that a specification is optional. Vertical bars "|" are used to indicate alternative expressions. For example, the rule

```
<!ELEMENT B (A, (X | C)?)>
<!ELEMENT X (LIST))

<!ELEMENT A EMPTY>
<!ATTLIST A d CDATA #REQUIRED>

<!ELEMENT C (#PCDATA)>
```

means that an element `B` is expanded into an "`A`" followed by an optional "`X`" or "`C`". The element `X` is expanded to another element "`LIST`". The element `A` does not contain any data between its start and end tags but has a mandatory character data attribute "`d`" that has to be specified. The element `C` contains Parsed Character Data (PCDATA) . The "#"words are XML keywords and should be used as is (case sensitive).

If an element (as in "`A`") is specified to be EMPTY, it means that there will not be anything between its start and end tags. The usage of such tags within a XML document can be in either of the following two ways:

`<A d="something"/>` or `<A d="something"></A>`

An element can also be defined as ANY, in which case, between the start and end tags of that element any other tag or data can be specified in the XML document.
Just as elements represent the document's logical structure, entities represent its physical structure. A reference to a declared entity can be done in the DTD. For example

```
<!ENTITY % inline "#PCDATA | EMPHASIS | LINK" >
<!ELEMENT PARA (%inline;)
```

is equivalent to

```
<!ELEMENT PARA (#PCDATA | EMPHASIS | LINK)>.
```

White space in the language follows the XML specification. White space, such as spaces, tabs and blank lines, is used to set apart the markup for greater readability. However, "significant" white space within character data is preserved. In XML, all markup tags are case sensitive and the XML documents using this

DTD should use the tags as defined. Thus a "WORK" tag is different from "work" tag or "wORk" tag. The XrML DTD defines all the element tags by capitalizing the names and all attribute tags in the lower case.

The comments in a XML document begins with "<!--" and ends with "-->" characters.
Design goals for XrML documents written in XML include

- *Simple parsing*. The XML is compatible with SGML and is easy to write programs which process XML documents.
- *Extensive defaulting*. Many parameters are optional. The mandatory parameters are specified and optional parameters have well-defined defaults.
- *Extensibility*. In the XrML DTD, it is easy to add new features unambiguously without substantially modifying the language or rewriting old specifications.

2.2 Top-Level Structure

XrML documents are applied to arbitrary digital content in a form of a single work, a work and its component parts, or a collection of works. A digital content is identified using an object. Various usage rights are specified for the content. The content identification and rights specification, together with their meta information (such as who issues this XrML document, when and to whom) are cryptographically signed to ensure their integrity. XrML documents are represented by the XrML element which has the following structure (where "?" denotes zero or one occurrence; "+" denotes one or more occurrences; and "*" denotes zero or more occurrences):

```
<XrML>
  <BODY>
        (ISSUED)?
        (TIME)?
        (DESCRIPTOR)?
        (ISSUER)?
        (ISSUEDPRINCIPALS)?
        (WORK)?
        (AUTHENTICATEDDATA)?
  </BODY>
  (SIGNATURE)?
</XrML>
```

Within the root element XrML, there are a mandatory element BODY and an optional element SIGNATURE; the latter is the digital signature of the former used to ensure its integrity. The BODY element consists of an optional description of a digital work and some optional meta information about this XrML document. The element TIME is a time interval in which this XrML document is valid. The element ISSUED is the time moment at which this document is issued. The element DESCRIPTOR is a description of this document. The element ISSUER is a principal who issues this XrML document. The element ISSUEDPRINCIPALS is a list of principals this document is issued to. The element WORK defines a digital work and its usage rights. Finally, the element AUTHENTICATEDDATA captures data that is provided to an application which processes this XrML document.

The top-level structure provides a level of flexibility in defining associations among ISSUER, ISSUEDPRINCIPALS and WORK based on their presence within a XrML document. If all the three elements are present, then the XrML document defines an association among the ISSUER, ISSUEDPRINCIPALS and WORK, namely, the ISSUER issues the XrML document to the ISSUEDPRICIPALS regarding usage rights on the WORK. If only the elements ISSUER and WORK are present, then the XrML document only establishes a relationship between the ISSUER and WORK, that is, the ISSUER issues the XrML document on usage rights of the WORK. This type of relationship is useful, for instance, for content owners to define usage rights before issued principals are known. It is those associations that can be used to meet different requirements of various situations.

The following example is an Activation License for Microsoft E-Books. It shows how to use top-level XrML structure without reference to any digital work. The example in the next subsection will show how a digital work and its usage rights are specified.

```
<!DOCTYPE XrML SYSTEM "xrml.dtd">
<XrML>
   <BODY type="LICENSE" version="2.0">
      <ISSUED>2000-01-27T15:30</ISSUED>
      <TIME>
          <FROM>2000-02-3T17:26</FROM>
          <UNTIL>3000-02-3T17:26</UNTIL>
      </TIME>
      <DESCRIPTOR>
         <OBJECT type="Registration-EUL">
            <ID type="GUID">6ae986ab-09a4-4210-8e4d-709dbd1e7c72</ID>
            <NAME>A Registration Certificate for John Doe</NAME>
         </OBJECT>
      </DESCRIPTOR>
      <ISSUER>
         <OBJECT type="Corporation">
            <ID type="Licensing Root">2</ID>
            <NAME>xyz.com</NAME>
            <ADDRESS type="URL">www.xyz.com</ADDRESS>
         </OBJECT>
         <PUBLICKEY>
            <ALGORITHM>RSA-512</ALGORITHM>
            <PARAMETER name="public exponent">
               <VALUE encoding="integer32">3</VALUE>
            </PARAMETER>
            <PARAMETER name="MODULUS">
               <VALUE encoding="base64" size="90">33845UR … </VALUE>
            </PARAMETER>
         </PUBLICKEY>
      </ISSUER>
      <ISSUEDPRINCIPALS>
         <PRINCIPAL internal-id="1">
            <OBJECT type="Device">
               <ID type="Registration MID 1.0">{47656e7 …}</ID>
               <NAME>John Doe's Computer</NAME>
            </OBJECT>
            <ENABLINGBITS type="sealed-private-key">
               <VALUE encoding="base64" size="2560"> GcsCCBXGV7vAI/0 …
               </VALUE>
               <PRINCIPAL internal-id="2"/>
            </ENABLINGBITS>
         </PRINCIPAL>
         <PRINCIPAL internal-id="2">
            <OBJECT type="Registration">
               <ID type="Registration ID">johnd</ID>
               <NAME>John Doe</NAME>
            </OBJECT>
            <PUBLICKEY>
               <ALGORITHM>RSA-512</ALGORITHM>
               <PARAMETER name="public exponent">
                  <VALUE encoding="integer32">65537</VALUE>
               </PARAMETER>
               <PARAMETER name="modulus">
                  <VALUE encoding="base64" size="512">hbJH0y … </VALUE>
               </PARAMETER>
            </PUBLICKEY>
         </PRINCIPAL>
```

```
      </ISSUEDPRINCIPALS>
   </BODY>
   <SIGNATURE>
      <DIGEST>
         <ALGORITHM>SHA1</ALGORITHM>
         <PARAMETER name="codingtype">
            <VALUE encoding="string">surface-coding</VALUE>
         </PARAMETER>
         <VALUE encoding="base64" size="160">OtSrhCKWHI … </VALUE>
      </DIGEST>
      <VALUE encoding="base64" size="512">A7qsNTFT2 … </VALUE>
   </SIGNATURE>
</XrML>
```

2.3 Digital Works and Their Usage Rights

XrML specifies a digital work and its usage rights in a similar way used in DPRL version 2. The overall structure for defining a work is as follows.

```
<WORK>
      (OBJECT)
      (DESCRIPTION)?
      (CREATOR)*
      (OWNER)?
      (DIGEST)*
      (PARTS)?
      (CONTENTS)?
      (COPIES)?
      (COMMENT)?
      (SKU)?
      (RIGHTSGROUP | REFERENCEDRIGHTSGROUP )+
</WORK>
```

The element `OBJECT` is the object that can be used to identify the work. The element `DESCRIPTION` is a description of the work. The element `CREATOR` is the creator of the work. The element `OWNER` is the owner of the work; it may or may not be same as the creator. The element `DIGEST` is cryptographic digest value of the work; it is used to ensure the integrity of the work. The element `PARTS` lists all parts of the work that may have different usage rights, fees and conditions; each part is a work itself. The element `CONTENTS` is an indicator of where the content is within a digital work; this is useful when the content covered by the usage rights is embedded within a digital work. The element `COPIES` is the number of copies of the work that are specified. The element `COMMENT` is a field for any comment on the digital work and its usage rights. The element `SKU` is a Stock Keeping Unit, which is used for extensibility to allow people to identify this work in their own stock. The element `RIGHTSGROUP` is a rights group that defines all usage rights associated with the work. The element `REFERENCEDRIGHTSGROUP` is a reference rights group of the work.

The following example shows how to specify a digital work and its usage rights. It also shows how to convert a rights label in DPRL (which does not have the XrML top-level features) to a XrML document.

```
<?xml version="1.0"?>
<!DOCTYPE XrML SYSTEM "xrml.dtd">
<XrML>
    <BODY type = "WORK" version="2.0">
        <WORK>
            <OBJECT type="BOOK-LIT-FORMAT">
                <ID type="ISBN">8374-39384-38472</ID>
                <NAME>A book of James</NAME>
          </OBJECT>
          <AUTHOR>James the first</AUTHOR>
```

```
    <AUTHOR>James the second</AUTHOR>
    <OWNER>
      <OBJECT type="Person">
          <ID type="US-SSN">103-74-8843</ID>
        <NAME>Mike the man</NAME>
        <ADDRESS type="email">mike@man.com</ADDRESS>
      </OBJECT>
  </OWNER>
    <PARTS>
        <WORK>
        <OBJECT type="Chapter">
            <ID type="relative">0</ID>
                <NAME>Chapter 1</NAME>
            </OBJECT>
      </WORK>
      <WORK>
        <OBJECT type="Image">
            <ID type="relative">1</ID>
                <NAME>Image 1: Photon Celebshots Dogs</NAME>
            </OBJECT>
      </WORK>
    </PARTS>
    <RIGHTSGROUP name="Main Rights">
        <DESCRIPTION>Rights granted to John Doe</DESCRIPTION>
        <BUNDLE>
            <TIME>
               <FROM>2000-01-27T15:30</FROM>
               <UNTIL>2000-01-27T15:30</UNTIL>
            </TIME>
            <ACCESS>
            <PRINCIPAL>
              <OBJECT type="Principal-Certificate">
                 <ID type="MS-GUID">7BD394EA … </ID>
                 <NAME>John Doe</NAME>
              </OBJECT>
              <ENABLINGBITS type="sealed-des-key">
                <VALUE encoding="base64" size="512">lnHtn …
                </VALUE>
              </ENABLINGBITS>
                </PRINCIPAL>
        </ACCESS>
    </BUNDLE>
        <RIGHTSLIST>
        <VIEW>
            <ACCESS>
              <PRINCIPAL>
                  <OBJECT type="MS Ebook Device">
                    <ID type="INTEL SN">
                       Intel PII 92840-AA9-39849-00
                    </ID>
                    <NAME>Johns Computer</NAME>
                </OBJECT>
                <AUTHENTICATOR type="xrml-certverify-program">
                    <ID type="microsoft-progid">abcd-93a1</ID>
                    <AUTHENTICATIONCLASS>
                        <VERSION>1.x-2.5</VERSION>
                    </AUTHENTICATIONCLASS>
                    <VERIFICATIONDATA type="authenticode-named-
root">
                        <PUBLICKEY>
                            <ALGORITHM>RSA-512</ALGORITHM>
```

```
                                    <PARAMETER name="public exponent">
                                        <VALUE encoding="integer32">
                                             65537
                                        </VALUE>
                                    </PARAMETER>
                                    <PARAMETER name="modulus">
                                        <VALUE encoding="base64"
                                            size="512"> u+aEb …
                                        </VALUE>
                                    </PARAMETER>
                                    </PUBLICKEY>
                                </VERIFICATIONDATA>
                                <VERIFICATIONDATA>
                                    <PARAMETER name="bbid">
                                      <VALUE encoding="string">xxzzy</VALUE>
                                  </PARAMETER>
                                  <PUBLICKEY>
                                      <ALGORITHM>RSA-512</ALGORITHM>
                                      <PARAMETER name="public exponent">
                                          <VALUE encoding="integer32">
                                                 3</VALUE>
                                      </PARAMETER>
                                      <PARAMETER name="modulus">
                                          <VALUE encoding="base64"size="90">
                                                 33845URT203987==</VALUE>
                                      </PARAMETER>
                                  </PUBLICKEY>
                                </VERIFICATIONDATA>
                            </AUTHENTICATOR>
                            </PRINCIPAL>
                      </ACCESS>
                    </VIEW>
                    <PRINT maxcount="5">
                        <FEE>
                          <MONETARY>
                              <PERUSE value="5.00">
                                <CURRENCY iso-code="USD"/>
                                    </PERUSE>
                                <ACCOUNT>
                                  <ACCOUNTFROM id="BA-0234-0928392"/>
                                  <HOUSE id="XYZ"
url="http://somehouse.com/payme.asp"/>
                                </ACCOUNT>
                            </MONETARY>
                      </FEE>
                      <TRACK>
                         <PRINCIPAL>
                            <OBJECT type="Tracking Server">
                               <ID type="Tracker ID">US1023</ID>
                               <NAME>e-tracker</NAME>
                               <ADDRESS type="url">
                                "http://sometrackingservice/trackme.asp"&gt;
                               </ADDRESS>
                            </OBJECT>
                            <VERIFICATIONDATA type="url">
                                <PARAMETER name="hashed-url">
                                    <VALUE encoding="base64" size="90">
                                        xxzzy</VALUE>
                                </PARAMETER>
                                <DIGEST>
                                  <ALGORITHM>SHA1</ALGORITHM>
```

```
                                <PARAMETER name="codingtype">
                                    <VALUE encoding="string">
                                        surface-coding</VALUE>
                                </PARAMETER>
                                <VALUE encoding="base64" size="160">
                                     OtSrhD5GrzxMeFEm8q4pQlCKWHI=</VALUE>
                              </DIGEST>
                              </VERIFICATIONDATA>
                           <PUBLICKEY>
                              <ALGORITHM>RSA-512</ALGORITHM>
                              <PARAMETER name="public exponent">
                                <VALUE encoding="integer32">3</VALUE>
                              </PARAMETER>
                              <PARAMETER name="MODULUS">
                                <VALUE encoding="base64" size="90">
                                    33845URT203987==
                                </VALUE>
                              </PARAMETER>
                            </PUBLICKEY>
                         </PRINCIPAL>
                         <PARAMETER name="tracking support address">
                             <VALUE encoding ="url">
                                "http://sometrackingservice/supportme.asp">
                           </VALUE>
                         </PARAMETER>
                     </TRACK>
                     <TERRITORY>
                         <LOCATION country="us" state="CA" city="El
Segundo" zipcode = "90245"/>
                         <LOCATION country="jp"/>
                     </TERRITORY>
                  </PRINT>
              </RIGHTSLIST>
           </RIGHTSGROUP>
       </WORK>
    </BODY>
</XrML>
```

3 TOP-LEVEL SYNTAX

The general structure of an XML signature is described in section 2: Overview and Examples. This section provides detailed syntax of the top-level features and actual examples. The syntax is defined via XML DTDs.

3.1 The XrML Element

The XrML element is the root element of a XrML document. A XrML document consists of a body and possibly a digital signature of the body.

DTD:

```
<!ELEMENT XrML (BODY, SIGNATURE?)>
<!ATTLIST XrML
  xrmlns    %URI;    #FIXED 'http://www.xrml.org/xrml'>
```

The attribute xrmlns is the name space for the latest XrML DTD.

3.2 The BODY Element

The BODY element consists of an optional description of a digital work and some optional meta information about this XrML document.

DTD:

```
<!ELEMENT BODY (ISSUED?,
                TIME?,
                DESCRIPTOR?,
                ISSUER?,
                ISSUEDPRINCIPALS?,
                WORK?,
                AUTHENTICATEDDATA?)>
<!ATTLIST BODY
  type      CDATA      #IMPLIED
  version   CDATA      #IMPLIED
  xrmlns    %URI;      #IMPLIED>
```

The element ISSUED is the time moment at which this document is issued. The element TIME is a time interval in which this XrML document is valid. The element DESCRIPTOR is a description of this document. The element ISSUER is a principal who issues this XrML document. The element ISSUEDPRINCIPALS is a list of principals this document is issued to. The element WORK defines a digital work and its usage rights. Finally, the element AUTHENTICATEDDATA captures data that is provided to an application which processes this XrML document.

The attributes type, version and xrmlns designate the body type, body version and schema. They are intended to facilitate backward compatibility to XrML documents that were written in an old version of the language. In addition, they enable customization of the XrML language through a body type to a specific subset of the XrML elements for a certain purpose.

Example:

```
<BODY type ="license" version="1.0" xrmlns="x-chema:license1Schema.xml">
```

Example:

```
<BODY type ="work" version="2.0" xrmlns="x-Schema:work2Schema.xml">
```

These two examples show that using different attributes of BODY makes it possible to customize XrML to a specific BODY definition such as license or work, as shown by the two examples in Section 2.

3.3 The ISSUED Element

The element ISSUED specifies the date and time when this XrML document is issued. It uses the format, yyyy-mm-ddThh:mm:ss, to specify year yyyy, month mm, day dd, hour hh, minute mm, and second ss. The time is specified in GMT (Greenwhich Mean Time).

```
DTD:

<!ELEMENT ISSUED (#PCDATA)>

Example:

<ISSUED>2000-01-27T15:30</ISSUED>
```

This example specifies that the issued time is at 30 minutes pass 3 o'clock in the afternoon on January 27, 2000.

3.4 The TIME Element

The element TIME is a time interval in which this XrML document is valid.

```
DTD:

<!ELEMENT TIME ( ( FROM     |
                   INTERVAL |
                   METERED)?,
                 UNTIL) >
Example:

<TIME>
      <FROM>2000-02-3T17:26</FROM>
      <UNTIL>3000-02-3T17:26</UNTIL>
</TIME>
```

This example shows that the valid time interval for this XrML document is from the time 17 o'clock 26 minutes of the day February 3rd, year 2000, to the time 17 o'clock 26 minutes of the day February 3rd, year 3000.

3.5 The DESCRIPTOR Element

The element DESCRIPTOR is a description of this document. It uses the generic element OBJECT as an identifier.

```
DTD:

<!ELEMENT DESCRIPTOR (OBJECT)>
```

3.5.1 The OBJECT Element

The element OBJECT is a generic entity for identifying some object. It consists of one mandatory element ID and two optional elements NAME and ADDRESS. The element ID is an identification of the object. The element NAME is the name of the object. The element ADDRESS specifies a location of the object.

```
DTD:

<!ELEMENT OBJECT (ID,
```

```
                  NAME?,
                  (ADDRESS*)?)>
<!ATTLIST OBJECT
  type           CDATA     #IMPLIED
  version        CDATA     #IMPLIED>

<!ELEMENT ID (#PCDATA)>
<!ATTLIST ID
  type           CDATA     #REQUIRED
  version        CDATA     #IMPLIED
  anonymitylevel CDATA     #IMPLIED>

<!ELEMENT NAME (#PCDATA)>

<!ELEMENT ADDRESS (#PCDATA)>
<!ATTLIST ADDRESS
  type           CDATA      #REQUIRED>
```

Example:

```
<OBJECT type="Registration-EUL">
      <ID type="GUID">6ae986ab-09a4-4210-8e4d-709dbd1e7c72</ID>
      <NAME>A Registration Certificate for John Doe</NAME>
      <ADDRESS type="IP">123.456.789</ADDRESS>
</OBJECT>
```

3.6 The ISSUER Element

The element ISSUER specifies the principal who issues this XrML document. It uses a generic entity aPrincipal to identify the principal (see the definition of the element PRINCIPAL). Its attribute type specifies the type of the ISSUER, and its attribute internal-id defines an ID number that is used within this XrML for reference to the ISSUER.

DTD:

```
<!ELEMENT ISSUER  (%aPrincipal;)>
<!ATTLIST ISSUER
  type          CDATA    #IMPLIED
  internal-id   CDATA    #IMPLIED>
```

Exmaple:

```
<ISSUER type="application" internal-id = "1">
      <OBJECT type="Corporation">
            <ID type="Licensing Root">2</ID>
            <NAME>xyz.com</NAME>
            <ADDRESS type="URL">www.xyz.com</ADDRESS>
      </OBJECT>
      <PUBLICKEY>
            <ALGORITHM>RSA-512</ALGORITHM>
            <PARAMETER name="public exponent">
               <VALUE encoding="integer32">3</VALUE>
            </PARAMETER>
            <PARAMETER name="MODULUS">
               <VALUE encoding="base64" size="90">33845UR … </VALUE>
            </PARAMETER>
      </PUBLICKEY>
</ISSUER>
```

The ISSUER in this example has type "application" and internal-id "1". It is identified by as a corporation xyz.com which is the licensing root 2 at the url www.xyz.com. It has an RSA public key whose components are the exponential 3 and modulus 33845UR ….

3.7 The ISSUEDPRICIPALS Element

The element ISSUEDPRINCIPALS specifies a list of principals to whom this XrML document is issued. See the definition of PRINCIPAL for details.

DTD:

```
<!ELEMENT ISSUEDPRINCIPALS (PRINCIPAL+)>
```

Example:

```
<ISSUEDPRINCIPALS>
      <PRINCIPAL internal-id="1">
            <OBJECT type="Device">
                  <ID type="Registration MID 1.0">{47656e7 …}</ID>
                  <NAME>John Doe's Computer</NAME>
            </OBJECT>
            <ENABLINGBITS type="sealed-private-key">
                  <VALUE encoding="base64" size="2560">GcsCCB … </VALUE>
                  <PRINCIPAL internal-id="2"/>
            </ENABLINGBITS>
      </PRINCIPAL>
      <PRINCIPAL internal-id="2">
            <OBJECT type="Registration">
                  <ID type="Registration ID">johnd</ID>
                  <NAME>John Doe</NAME>
            </OBJECT>
            <PUBLICKEY>
                  <ALGORITHM>RSA-512</ALGORITHM>
                  <PARAMETER name="public exponent">
                        <VALUE encoding="integer32">65537</VALUE>
                  </PARAMETER>
                  <PARAMETER name="modulus">
                        <VALUE encoding="base64" size="512">
                              hbJH0y … </VALUE>
                  </PARAMETER>
            </PUBLICKEY>
      </PRINCIPAL>
</ISSUEDPRINCIPALS>
```

3.8 The PRINCIPAL Element

The element PRINCIPAL defines a generic, unnamed entity for a principal. It has optional attributes type and internal-id. The internal-id attribute is used as a reference to this principal within the XrML document.

DTD:

```
<!ELEMENT PRINCIPAL  (%aPrincipal;) >
<!ATTLIST PRINCIPAL
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED>
```

See the element ISSUEDPRINCIPALS for examples of principals.

3.8.1 The `aPrincipal` Entity

The `aPrincipal` entity specifies the mandatory and optional component elements for defining a principal. As shown in the following DTD, it consists of an optional element `OBJECT` to identify a principal, and zero or more occurrences of elements `AUTHENTICATOR`, `PUBLICKEY`, `PRIVATEKEY`, `DIGEST`, `VERIFICATIONDATA`, `ENABLINGBITS`, and `CERTIFICATE`.

The element `AUTHENTICATOR` is a principal that authenticates this principal. The laments `PUBLICKEY` and `PRIVATEKEY` are a public key and a private key of this principal. The element `DIGEST` is a cryptographic digest value needed to ensure the identity of this principal. The element `VERIFICATIONDATA` is some information that the authenticator needs to verify the identity of this principal. The element `ENABLINGBITS` is some extra (secret) information associated with this principal. The element `CERTIFICATE` is a reference to a digital certificate of this principal. The element `PRINCIPAL` is an optional principal whose credential and keys are used to seal and unseal the element `ENABLINGBITS`. The element `SECURITYLEVEL` is an optional element to define a security level of the principal.

When more than one occurrences of each optional element are present, any one of them can be used for its purpose. For instance, if there are more than one `AUTHENTICATOR` elements, then any one of them can be used to validate the identity of this principal.

`DTD:`

```
<!ENTITY % aPrincipal
  "OBJECT?,
   (AUTHENTICATOR    |
    PUBLICKEY        |
    PRIVATEKEY       |
    DIGEST           |
    VERIFICATIONDATA |
    ENABLINGBITS     |
    CERTIFICATE      |
    PRINCIPAL        |
    SECURITYLEVEL)*" >
```

See Section 4 for the definition of the element `CERTIFICATE`.

3.8.2 The `AUTHENTICATOR` Element

The element `AUTHENTICATOR` is a principal who authenticates other principals. It consists of a mandatory element `ID` and optional elements `NAME`, `AUTHENTICATOR`, `AUTHENTICATIONCLASS`, and `VERIFICATIONDATA`. The elements ID and NAME are the id and name of this authenticator. The element `AUTHENTICATOR` is an authenticator who authenticates this authenticator; With this element, one can build an authentication chain. The element `AUTHENTICATIONCLASS` is a classification of the authenticator. The element `VERIFICATIONDATA` is some information the element `AUTHENTICATOR` is going to use to authenticate this authenticator. There could be multiple `VERIFICATIONDATA` elements, and when that happens, any one of them can be used.

`DTD:`

```
<!ELEMENT AUTHENTICATOR (ID,
                         NAME?,
                         AUTHENTICATOR?,
                         AUTHENTICATIONCLASS?,
                         VERIFICATIONDATA+)>
<!ATTLIST AUTHENTICATOR
  type          CDATA     #REQUIRED
  internal-id   CDATA     #IMPLIED>
```

Example:

```
<AUTHENTICATOR type="xrml-certverify-program">
      <ID type="microsoft-progid">abcd-93a1</ID>
      <AUTHENTICATIONCLASS>
            <VERSION>1.x-2.5</VERSION>
      </AUTHENTICATIONCLASS>
      <VERIFICATIONDATA type="authenticode-named-root">
            <PUBLICKEY>
                  <ALGORITHM>RSA-512</ALGORITHM>
                  <PARAMETER name="public exponent">
                        <VALUE encoding="integer32">65537</VALUE>
                  </PARAMETER>
                  <PARAMETER name="modulus">
                        <VALUE encoding="base64" size="512"> u+aEb …
                        </VALUE>
                  </PARAMETER>
            </PUBLICKEY>
      </VERIFICATIONDATA>
      <VERIFICATIONDATA>
            <PARAMETER name="bbid">
                  <VALUE encoding="string">xxzzy</VALUE>
            </PARAMETER>
            <PUBLICKEY>
                  <ALGORITHM>RSA-512</ALGORITHM>
                  <PARAMETER name="public exponent">
                        <VALUE encoding="integer32">3</VALUE>
                  </PARAMETER>
                  <PARAMETER name="modulus">
                        <VALUE encoding="base64" size="90">
                              33845URT203987==
                        </VALUE>
                  </PARAMETER>
            </PUBLICKEY>
      </VERIFICATIONDATA>
</AUTHENTICATOR>
```

3.8.3 The AUTHENTICATIONCLASS Element

The element AUTHENTICATIONCLASS is a classification of the authenticator. It has the component elements VERSION, VERSIONSPAN and SECURITYLEVEL. The VERSION element specifies the current version. The VERSIONSPAN element covers all previous versions of the AUTHENTICATIONCLASS element. The element SECURITYLEVEL defines a security level of this authentication class.

DTD:

```
<!ELEMENT AUTHENTICATIONCLASS ( (VERSION     |
                                 VERSIONSPAN |
                                 SECURITYLEVEL)*)>

<!ELEMENT VERSION (#PCDATA)>

<!ELEMENT VERSIONSPAN EMPTY>
<!ATTLIST VERSIONSPAN
  min            CDATA     #IMPLIED
  max            CDATA     #IMPLIED>
```

For the DTD definition of the element SECURITYLEVEL, see Section 4.

See the element AUTHENTICATOR for an example.

3.8.4 The VERIFICATIONDATA Element

The element VERIFICATIONDATA is some information that the authenticator needs to verify the identity of this principal.

DTD:

```
<!ELEMENT VERIFICATIONDATA ( (PUBLICKEY |
                              PRIVATEKEY |
                              PARAMETER |
                              DIGEST)*)>
<!ATTLIST VERIFICATIONDATA
  type          CDATA     #REQUIRED>
```

See the element AUTHENTICATOR for an example.

3.8.5 The PUBLICKEY and PRIVATEKEY Elements

The laments PUBLICKEY and PRIVATEKEY are a public key and a private key of a principal. They have component elements ALGORITHM and PARAMETER. These elements are used to identify the encryption or signature algorithm and necessary parameters that these keys are associated with.

DTD:

```
<!ELEMENT PUBLICKEY (ALGORITHM,
                     PARAMETER*)>

<!ELEMENT PRIVATEKEY (ALGORITHM,
                      PARAMETER*)>

<!ELEMENT ALGORITHM (#PCDATA)>

<!ELEMENT PARAMETER (VALUE)>
<!ATTLIST PARAMETER
  name          CDATA     #REQUIRED>

<!ELEMENT VALUE (#PCDATA)>
<!ATTLIST VALUE
  encoding      CDATA     #REQUIRED
  size          CDATA     #IMPLIED>
```

See the element AUTHENTICATOR for an example.

3.8.6 The DIGEST Element

The element DIGEST is a cryptographic digest value needed to ensure the identity of a principal or the integrity of an object. It specifies a digest algorithm, its parameters, an optional digest value, and optionally the source data object being digested. The ALGORITHM element specifies an algorithm for generating the digest. The PARAMETER element specifies a parameter used by the digest algorithm. The optional VALUE element is the encoded digest value. The attribute sourcedata indicates the source data the digest value is generated from. When the attribute is absent, the source data is determined from the context. For instance, if this DIGEST element is part of the SIGNATURE element for a XrML document, the source data is the element BODY of the document.

DTD:

```
<!ELEMENT DIGEST (ALGORITHM,
                  PARAMETER*,
                  VALUE?)>
```

```
<!ATTLIST DIGEST
  sourcedata    CDATA     #IMPLIED>
```

See the element AUTHENTICATOR for an example.

3.8.7 The ENABLINGBITS Element

The element ENABLINGBITS is some extra (secret) information associated with a principal. It can be used to carry the information that enables a principal to use a XrML document or any other information (such as a work) it is associated with. It has one mandatory element VALUE. The element VALUE is the value of the enabling bits. By default, the value is sealed using the key of the principal that contains the ENABLINGBITS. However, if the containing principal contains another principal, then the key of the second principal is used to sealed and unseal the value of the ENABLINGBITS.

DTD:

```
<!ELEMENT ENABLINGBITS (VALUE)>
<!ATTLIST ENABLINGBITS
  type          CDATA     #REQUIRED>
```

Example:

```
<ENABLINGBITS type="sealed-private-key">
      <VALUE encoding="base64" size="2560">GcsCCBXGV7vAI/0 … </VALUE>
</ENABLINGBITS>
```

This example shows an enabling bits value.

3.9 The AUTHENTICATEDDATA Element

The element authenticateddata is an optional element that captures data that is necessary for an application which processes a XrML document.

DTD:

```
<!ELEMENT AUTHENTICATEDDATA (#PCDATA)>
<!ATTLIST AUTHENTICATEDDATA
    size CDATA  #REQUIRED
    name CDATA  #REQUIRED
  >
```

3.10 The SIGNATURE Element

The SIGNATURE element contains digital signature information of the BODY element in a XrML document. It contains mandatory elements DIGEST and VALUE and optional elements ALGORITHM and PARAMETER. The element DIGEST provides digest information of the BODY. The element ALGORITHM specifies the algorithm used for generating and validating the digital signature. The element PARAMETER defines parameters needed by the signature algorithm. The element VALUE is the actual value of the digital signature. It is encoded according to the identifier specified in the encoding attribute. Base64 is a commonly used encoding method. When the element ALGORITHM is absent, the algorithm associated with the public and/or private keys of the element ISSUER in the BODY is selected.

DTD:

```
<!ELEMENT SIGNATURE (DIGEST,
                     ALGORITHM?,
                     PARAMETER*,
                     VALUE)>
```

The definitions of the elements DIGEST, ALGORITHM, PARAMETER, and VALUE are defined in the previous subsection.

Example:

```
<SIGNATURE>
      <DIGEST>
         <ALGORITHM>SHA1</ALGORITHM>
         <PARAMETER name="codingtype">
            <VALUE encoding="string">surface-coding</VALUE>
         </PARAMETER>
         <VALUE encoding="base64" size="160">OtSrhCKWHI … </VALUE>
      </DIGEST>
      <VALUE encoding="base64" size="512">A7qsNTFT2 … </VALUE>
   </SIGNATURE>
```

4 WORK AND RIGHTS SYNTAX

This section specifies the XrML syntax for a work and its usage rights. Major part of this section is from the definition of DPRL version 2.0. The difference between XrML and DPRL version 2 is listed in Section 5.

4.1 Specifying a Work

The language design goal for a work specification is to provide a place for the information relevant to a digital work. A composite work would have a separate work specification for each part of the overall work.

DTD:

```
<!ELEMENT WORK (OBJECT,
                DESCRIPTION?,
                CREATOR*,
                OWNER?,
                DIGEST*,
                PARTS?,
                CONTENTS?,
                COPIES?,
                COMMENT?,
                SKU?,
                (RIGHTSGROUP | REFERENCEDRIGHTSGROUP)+) >

<!-- For definition of OBJECT, see Section 3. -->

<!ELEMENT DESCRIPTION (#PCDATA)>

<!ELEMENT CREATOR (#PCDATA)>
<!ATTLIST CREATOR
  type         CDATA      #IMPLIED>

<!ELEMENT OWNER ( %aPrincipal; )>
<!ATTLIST OWNER
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED>

<!ELEMENT PARTS (WORK+)>

<!-- For definition of DIGEST, see Section 3. -->

<!ELEMENT CONTENTS EMPTY>
<!ATTLIST CONTENTS
  from          CDATA    #REQUIRED
  to            CDATA    #REQUIRED>

<!ELEMENT COPIES EMPTY>
<!ATTLIST COPIES
  maxcount     CDATA   #IMPLIED>

<!ELEMENT COMMENT (#PCDATA)>

<!ELEMENT SKU (#PCDATA)>
<!ATTLIST SKU
  type         CDATA    #IMPLIED>
```

A work specification includes the element `OBJECT` that identifies the digital object of the work. A part of the element `OBJECT` is an element `ID`, which is a unique identification number to the digital work. Examples of such an identification number include an ISBN number for a book and an ISSN number for a

periodical. Assumptions and methods for identifying and rating works are discussed elsewhere in this document.

A work specification also includes an optional text description, DESCRIPTION, of the work for a variety of other information for identifying a work, but whose format is not necessarily fixed. The format and scope of the information in the text description is not specified, so that different organizations can put different information here. In general, the information in the text description is intended for human use. It is not used in the automatic rights processing of a repository.

A work specification includes an optional element CREATOR for the creator of the work. The creator has an attribute, type. It is used to characterize the creator. Possible values for the type attribute include "author" for articles and novels, "composer" for songs, and "photographer" for pictures.

The ability to change the rights on a work is not the same issue as adding further fees or restrictions when a work is included in a composite work. Rather, it is the right to change the terms and conditions within the work specification. A work specification includes an optional owner specification, OWNER, which indicates who is authorized to change any part of the work-specification, including adding or deleting rights, changing conditions, or raising or lowering fees. The owner is specified as a generic principal. The principal specification indicates what credentials of the owner are required to make the changes. If no owner specification is specified, then the issuer of this XrML document can make changes to the rights on a digital work. If the issuer specification is also absent, then anyone can make changes.

In many cases the originality and integrity of a digital work is very important and needs to be preserved. A work specification includes an optional element DIGEST to capture a cryptographic digest value of the work. When signed together with the rest of the XrML body, this value allows anyone to validate the association of the XrML document with the work and whether or not the integrity of the work has been compromised.

For composite works, the optional PARTS specification specifies a list of works which are included as parts of this work. The right specifications given in the higher level work specification are for the work as a whole. (They do not override the rights on individual included works. See the appendix on transaction semantics.)

The CONTENTS specification gives the starting and stopping addresses to which the rights in the work specification apply. If no contents specification is given, the work is interpreted as covering the entire attached digital work. When works are nested, the addresses are taken as relative to the beginning (origin) of the larger containing work. The form of an address depends on the medium on which the work is stored.

An optional COPIES specification gives the number of copies of the digital work. This field is referenced in the operation of various rights. If no copies specification is given, the number of copies of the digital work is taken to be one. If there are two or more copies, then it is possible to transfer or loan a copy of the work while exercising other rights on remaining copies.

A work specification includes an optional COMMENT. Comments in XrML are intended for documentation, typically used by people creating and updating rights specifications. Comments are not interpreted for meaning by trusted systems. However, persistence of comments is important in order for comments to serve their purpose. Comments have fixed and specific locations in the syntax (“structured comments”). There can be at most one comment in a work specification. Comments can also be used in other places in a specification, such as in rights groups, as described later.

A work specification also includes an optional element SKU. It stands for Stock Keeping Unit. It is included mainly for extensibility to allow a content retailer or distributor to identify a version of their stock. For example, it could be used to identify a particular set of symmetric keys used to encrypt books, or some other information that can be mapped to when a XrML document is presented again to the retailer or distributor.

Finally, a work specification includes a list of named groups of rights.

```
<?xml version="1.0"?>
<!DOCTYPE XrML SYSTEM "xrml.dtd">
<XrML>
    <BODY type = "WORK" version="2.0">
        <WORK>
            <OBJECT type="BOOK-LIT-FORMAT">
                <ID type="ISBN">8374-39384-38472</ID>
                <NAME>Zuke-Zack, the Moby Dog Story</NAME>
            </OBJECT>
            <DESCRIPTION>Copyright 1994 Jones Publishing</Description>
            <AUTHOR>John Beagle</AUTHOR>
            <OWNER>
              <OBJECT type="Person">
                  <ID type="US-SSN">103-74-8843</ID>
                <NAME>Mike the man</NAME>
                <ADDRESS type="email">mike@man.com</ADDRESS>
              </OBJECT>
              <CERTIFICATE>
                  <AUTHORITY id="Library of Congress"></AUTHORITY>
                  <PROPERTYPAIR name="ID" value="Murphy Publishers"/>
                    </CERTIFICATE>
            </OWNER>
            <PARTS>
                <WORK>
                    <OBJECT type="Chapter">
                        <ID type="relative">0</ID>
                        <NAME>Chapter 1</NAME>
                    </OBJECT>
                </WORK>
                <WORK>
                    <OBJECT type="Image">
                        <ID type="relative">1</ID>
                        <NAME>Image 1: Photon Celebshots Dogs</NAME>
                    </OBJECT>
                </WORK>
            </PARTS>
            <CONTENTS from="1" to="16636"/>
            <COMMENT>"Rights edited by Pete Jones, June 1996."</COMMENT>
            <RIGHTSGROUP name="Regular">
                  <COMMENT>This set of rights is used
                        for standard retail editions. </COMMENT>
            <BUNDLE>
                  <TIME>
                        <UNTIL>1998-01-01</UNTIL>
                  </TIME>
                  <ACCESS>
                        <SECURITYLEVEL name="Protocol" value="5"/>
                        <SECURITYLEVEL name="Physical" value="6"/>
                  </ACCESS>
                  <FEE><MONETARY><ACCOUNT>
                        <ACCOUNTTO id="123456789"/><HOUSE id="Visa"/>
                  </ACCOUNT></MONETARY></FEE>
            </BUNDLE>
            <RIGHTSLIST>
                  <PLAY>
                        <FEE><MONETARY><METEREDFEE>
                              <RATE value=".05"/><PER hours="1"/>
                              <BY seconds="1"/>
                        </METEREDFEE></MONETARY></FEE>
                  </PLAY>
                  <PRINT>
```

```
                        <PRINTER>
                              <CERTIFICATE>
                                    <AUTHORITY id="DPT"></AUTHORITY>
                                    <PROPERTYPAIR name="Type"
                                     value="TrustedPrinter-6"/>
                              </CERTIFICATE>
                        </PRINTER>
                        <WATERMARK>
                              <WATERMARK-STR string="Title: 'Zeke Zack
                  the Moby Dog' Copyright 1994 by Zeck Jones. All Rights
                  Reserved."/>
                        <WATERMARK-TOKENS all-rights="false"
                              render-rights="true"/>
                        <WATERMARK-STR string="Title: 'Zeke Zack the
                  Moby Dog' Copyright 1994 by Zeck Jones. All Rights
                  Reserved."/>
                        </WATERMARK>
                        <FEE><MONETARY><PERUSE value="10"/>
                        </MONETARY></FEE>
                  </PRINT>
                  <TREANSFER></TREANSFER>
                  <COPY>
                        <FEE><MONETARY><PERUSE value="10"/>
                        </MONETARY></FEE>
                  </COPY>
                  <COPY>
                        <ACCESS>
                              <PRINCIPAL type="user">
                                    <CERTIFICATE>
                                      <AUTHORITY id="Murphy Publishers">
                                      </AUTHORITY>
                                      <PROPERTYPAIR name="Type"
                                          value="Distributor"/>
                                    </CERTIFICATE>
                              </PRINCIPAL>
                        </ACCESS>
                  </COPY>
                  <DELETE></DELETE>
                  <BACKUP></BACKUP>
                  <RESTORE>
                        <FEE><MONETARY><PERUSE value="5"/>
                        </MONETARY></FEE>
                  </RESTORE>
            </RIGHTSLIST>
          </RIGHTSGROUP>
        </WORK>
     </BODY>
  </XrML>
```

This example of a work specification in XrML has one rights group. The work specification expresses conditions and fees for several rights: play, print, transfer, copy, delete, backup, and restore. The syntax, meaning and defaults of these various right specifications are explained later. The work includes two other parts, Chapter 1 and an image incorporated from other sources.

Usage rights specifications are organized in groups of rights. Various rights specify what usage rights are passed along when a work is copied, transferred, loaned, or used in derivative works. It is convenient to refer to groups of rights by name. Within a work specification, each rights group must have a unique name.

DTD:

```
<!ELEMENT RIGHTSGROUP (COMMENT?, BUNDLE?, RIGHTSLIST)>
<!ATTLIST RIGHTSGROUP
      name CDATA #REQUIRED>

<!ELEMENT REFERENCEDRIGHTSGROUP (COMMENT?, BUNDLE?, RIGHTSLIST)>
<!ATTLIST REFERENCEDRIGHTSGROUP
      name CDATA #REQUIRED>
```

There are two kinds of rights groups: regular rights groups and reference rights groups. A regular rights group contains a group of rights applied to the current work. A reference rights group contains a group of rights that can be referred to by name in transport rights and applied to subsequent copies of the work. A rights group can include an optional comment before the list of rights.

A rights group can include an optional bundle specification, which specifies time specifications, fee specifications, or access specifications that apply to all of the rights in the group. The precise meanings of time specifications, fee specifications, access specifications, and watermark specifications as they appear in bundle specifications are explained in later sections. A bundle specification can include an optional comment.

The rights for a digital work are explicitly listed. Any right not in the list is not granted. The name of the right identifies the kind of right. Kinds of rights fall into various classes as explained later.

There can be more than one instance of a particular kind of right for a given work. For example, there can be more than one instance of a `COPY` right, which grants permission to make new copies of a digital work. One instance of the `COPY` right may have a special rate for "book of the month club" members. Another instance of the `COPY` right may allow copies to made without charge if the consumer has an appropriate digital ticket.

DTD:

```
<!ELEMENT RIGHTSLIST ((COPY | TRANSFER | LOAN |
                       PLAY | PRINT | EXPORT | VIEW |
                       EDIT | EXTRACT | EMBED |
                       BACKUP | RESTORE | VERIFY | FOLDER | DIRECTORY |
                       DELETE | INSTALL | UNINSTALL)+)>
```

Each right, identified by its name, has an optional `COMMENT`, an optional `TIME` specification, an optional `ACCESS` specification, an optional `FEE` specification, an optional `TERRITORY` specification and an optional `TRACK` specification. Other than `COMMENT`, the others are considered as an XML entity, `termsConditions`.

DTD:

```
<!ENTITY % termsConditions
  "(TIME | ACCESS | FEE | TERRITORY | TRACK)+" >
```

All the rights being granted for the work is specified in the list. The various kinds of rights covered by the usage rights system are described in the rest of this section. The time specification states the period of time over which the right is granted. The access specification states conditions on the exercise of a right, such as a requirement for various authorizations. The fee specification states fees associated with the exercise of a right. The territory specification states at which location the right can be exercised. The track specification states where and how to track the usage. These specifications are described individually in the following subsections of this section.

The specifications for a given right are a combination of (1) the specifications explicitly within the right statement itself, and (2) specifications from an optional bundle specification in the rights group to which it belongs. If there is no time specification, then there is no time limit on the use of this right. If there is no

access specification, then there is no access test (including security level) required. If there is no fee specification, then there is no fee required for use. A right specification can include an optional comment.

4.2 Digital Property Rights

Rights belong to one of several categories. The rights in XrML describe uses that:

- Relate meaningfully to distinctions of use described in copyright law.
- Respect distinctions of use relevant to common licensing arrangements in commercial purposes of digital works.
- Are enforceable by practical trusted systems.
- Are exercised by carrying out a particular transaction on a repository.

Logically, rights can be categorized into Transport, Render, Derivative work, File management and rights for Configuration. Transport rights govern the movement of a work from one repository to another. Render rights govern the printing and display of a work, or more generally, the transmission of a work through a transducer to an external medium. (This includes the Export right, which can be used to make copies in the clear.) Derivative work rights govern the reuse of a work in creating new works. File management rights govern such things as making and restoring backup copies. Finally, configuration rights refer to the installation of software in repositories.

The next five subsections describe each of the rights that fall under each category.

4.2.1 Transport Rights

Transport rights govern the creation and movement of persistent copies of a work under the control of trusted repositories. There are three distinct kinds of transport rights: copy, transfer, and loan. The interpretation of these rights are similar to familiar operations on physical works: copying an audio tape, transferring (or giving someone) a book, or loaning a compact disc.

The design goals of the transport rights are to provide for the following:

- To distinguish between rights for moving works among repositories and rights that increase the total number of copies of a digital work.
- To describe controlled loaning of works, both personal loans and also institutional or library loans.
- To support a range of distribution models, including both licensed distributor models and consumer-based distribution (also known as super distribution).
- To enable publishers to specify how the rights change on new copies of works.
- To control what rights move to the loaner copy when a copy of a work is loaned out and also what rights stay behind.

DTD:

```
<!ELEMENT COPY (NEXTRIGHTS?, COMMENT?, (%termsConditions;)?)>
<!ELEMENT TRANSFER (NEXTRIGHTS?, COMMENT?, (%termsConditions;)?)>
<!ELEMENT LOAN (REMAININGRIGHTS?, NEXTRIGHTS?,
                COMMENT?, (%termsConditions;)?)>
```

A COPY right is the right to make a new digital copy of a work. (PRINT rights, described later, govern the making of hard copies.) The COPY right is invoked whenever a new digital copy is made. Like all rights, the right to copy can be parameterized by time, access, fee, and control specifications. An optional NEXTRIGHTS specification can be used to establish rights for the new copy that are distinct from the rights on the original.

A TRANSFER right is the right to transfer the digital work from one repository to another. A TRANSFER right does not increase the number of copies of a work because the transfer transaction between two trusted systems removes the digital work from the sending repository when the copy has been created and verified on the receiving repository.

A LOAN right is the right to loan a copy of the work for a period of time. A LOAN right creates a "loaner" copy of a work on a receiving repository. Typically, during the time that a work is loaned, the original copy of the work cannot be rendered. However, if there is more than one copy of the work on the original repository, then the original repository can still exercise all rights on copies that are not loaned out. As discussed below, exactly what rights are usable by either the loaning repository or at the loanee repository can be specified explicitly using the NEXTRIGHTS and the REMAININGRIGHTS specification.

Throughout the loan period, both repositories must keep track of the fact that the copy has been loaned and take this information into account in all relevant transactions. At the end of the loan period, the copy of the work on the receiving repository deactivates and the copy on the loaning repository reactivates.

DTD:

```
<!ELEMENT NEXTRIGHTS (RIGHTSTOADD | RIGHTSTODELETE)>
<!ELEMENT REMAININGRIGHTS EMPTY>
<!ATTLIST REMAININGRIGHTS
  name    CDATA    #REQUIRED>
```

When a digital work is copied, transferred, or loaned, certain rights become available on the receiving repository. Exactly which rights are available is determined by an optional NextCopyRights specification. By default, the rights on the new repository are the same as the rights on the original repository. However, the list of rights can be modified by specifically adding or deleting groups of rights. The groups named can be either regular rights groups or reference rights groups.

The rights specification for the next copy can be described as a set of differences from the original specification. These differences may be rights to add to those of the original and rights to delete from those of the original. The set of rights provided in the RIGHTSTOADD specification is added to those of the original work. The set of rights provided in the RIGHTSTODELETE specification is deleted from those of the original.

DTD:

```
<!ELEMENT RIGHTSTOADD EMPTY>
<!ATTLIST RIGHTSTOADD name    CDATA    #REQUIRED>

<!ELEMENT RIGHTSTODELETE EMPTY>
<!ATTLIST RIGHTSTODELETE name    CDATA    #REQUIRED>
```

The optional REMAININGRIGHTS specification describes what rights to keep behind in the original loaned work. If this specification is not given, then no rights are retained when the last copy is loaned out. Here is a simple example of a LOAN usage right that specifies what rights to keep and which ones to give on the loaned copy. In this case, the loaner copy gets all of the rights except the book club rights, which are retained, by the original repository.

Example:

```
<LOAN>
   <NEXTRIGHTS><RIGHTSTODELETE name="Book-Club-Rights"/></NEXTRIGHTS>
   <REMAININGRIGHTS name="Book-Club-Rights"/>
   <ACCESS><SECURITYLEVEL></SECURITY></ACCESS>
   <ACCESS>
       <SECURITYLEVEL name="Physical" value="4"/>
       <SECURITYLEVEL name="Kernal" value="5"/>
   </ACCESS>
</LOAN>
```

Here is another example of a work specification. This work specification has two rights groups, one called “Distributor” and one called “Consumer.”

Example:

```
<?XML version="1.0" encoding="UTF-8"?>
<!DOCTYPE Work SYSTEM "xrml.dtd">
<XrML>
  <WORK>
    …
      <RIGHTSGROUP name="Distributor">
        <COMMENT>"These rights limited to our licensed distributors."
        </COMMENT>
        <BUNDLE>
            <ACCESS>
                  <SECURITYLEVEL name="Physical" value="4"/>
                  <SECURITYLEVEL name="Kernel" value="5"/>
                  <PRINCIPAL type="User"><CERTIFICATE>
                        <AUTHORITY id="Murphy Publishers">
                        </AUTHORITY>
                        <PROPERTYPAIR name="Type" value="Distributor"/>
                        </CERTIFICATE></PRINCIPAL>
            </ACCESS>
        </BUNDLE>
      <RIGHTSLIST>
        <COPY>
           <FEE><TICKET type="Inventory-987>
                  <AUTHORITY id="Murphy Publishers"/>
           </TICKET></FEE>
        </COPY>
        <PLAY></PLAY>
      </RIGHTSLIST>
    </RIGHTSGROUP>
    <RIGHTSGROUP name="Consumer">
        <COMMENT>"These rights exercised by any owner of our works."
        </COMMENT>
        <BUNDLE>
            <ACCESS>
                <SECURITYLEVEL name="Physical" value="4"/>
                <SECURITYLEVEL name="Kernel" value="5"/>
            </ACCESS>
            <FEE>
                <MONETARY><ACCOUNT>
                  <ACCOUNTTO id="123456789"/><HOUSE id="Visa"/>
                </ACCOUNT></MONETARY>
            </FEE>
        </BUNDLE>
        <RIGHTSLIST>
            <COPY><NEXTRIGHTS><RIGHTSTODELETE name="Distributor"/>
                  </NEXTRIGHTS>
                  <FEE><MONETARY><PERUSE value="10"/>
                    <ACCOUNT>
                        <ACCOUNTTO id="Id-678-qwerqeruyt"/>
                  </ACCOUNT></MONETARY></FEE>
            </COPY>
            <PLAY>
                <FEE><MONETARY><METEREDFEE>
                        <RATE value=".05"/><PER hours="1"/>
                </METEREDFEE></MONETARY></FEE>
            </PLAY>
            <DELETE><COMMENT>"This right is without restrictions."
```

```
                    </COMMENT>
                </DELETE>
                <TREANSFER></TREANSFER>
                <LOAN>
                    <REMAININGRIGHTS name="Distributor"/>
                    <NEXTRIGHTS><RIGHTSTODELETE name="Distributor"/>
                    </NEXTRIGHTS>
                </LOAN>
            </RIGHTSLIST>
        </RIGHTSGROUP>
    </WORK>
<XrML>
```

The distributor rights include the right to copy a work without fee except that a particular digital ticket must be punched. The distributor rights also include a right to play the work without fee. Each of the rights in the "Distributor" rights group requires a Murphy's distributor digital license to exercise.

There is also a "Consumer" rights group. This group includes rights to copy, play, delete, transfer, and loan. The copy right has a set fee to make a copy and the play right has a metered charge of five cents per hour. A loaner copy does not carry the distributor rights.

4.2.2 Render Rights

Render rights govern the creation of representations of a digital work outside of the control of trusted systems.

According to the dictionary, to render something is to represent it in a verbal form or in a drawing or a painting. In the context of usage rights, we use the word "render" to refer to processes that transform a digital copy of a work into a form where it can be seen or used outside of the repository. For example, a digital book or digital photograph may be displayed on a screen, a piece of music may be played through a loud speaker, a movie may be shown on a screen, or a page of a book may be printed on a printer.
When a digital work is rendered, information moves outside of the control of a trusted system. This presentation of the work as image or sound makes it possible for someone to see or hear the work. Necessarily, it also makes the work accessible for analog copying. Anything you can see you can photograph; anything you can hear you can record. Rendering does not imply permission to make other recordings of the work, such as with a film camera, video camera, or tape recorder, even though such copies will typically have lower quality than the digital original. The right to make such copies is governed by copyright law. Many techniques for rendering from digital originals also allow the inclusion of information in watermarks and fingerprints, making it possible to identify the work and also the person who played it. Although recordings of rendered works cannot be prevented, such non-digital copying of a work is governed by applicable copyright laws.

The design goals of the render rights are:

- To distinguish between rights for making ephemeral copies such as display images and rights for making long-lasting copies such as printouts.
- To provide a means of distinguishing different kinds of players and printers for digital works such as ones that add distinguishing codes. For example, a publisher may want to limit printing of a digital work to certain kinds of printers that are capable of embedding specific kinds of watermarks.
- To provide a controlled means for creating unencumbered digital copies outside of a trusted system.
- To provide a means for specifying information to be embedded in watermarks for works displayed or printed. Watermark data can be pre-specified by a publisher or distributor. It includes data that is not known until the time that a work is printed or displayed.

DTD:

```
<!ELEMENT PLAY (PLAYER?, COMMENT?, (%termsConditions;)?, WATERMARK?)>
<!ELEMENT PRINT (PRINTER?, COMMENT?, (%termsConditions;)?, WATERMARK?)>
<!ATTLIST PRINT maxcount CDATA #IMPLIED >
<!ELEMENT EXPORT (REPOSITORY?, COMMENT?, (%termsConditions;)?,
                  WATERMARK?)>
<!ELEMENT VIEW (VIEWER?, COMMENT?, (%termsConditions;)?, WATERMARK?)>

<!ELEMENT PLAYER ( %aPrincipal; )>
<!ATTLIST PLAYER
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED>

<!ELEMENT PRINTER ( %aPrincipal; )>
<!ATTLIST PRINTER
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED>

<!ELEMENT VIEWER ( %aPrincipal; )>
<!ATTLIST VIEWER
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED>

<!ELEMENT REPOSITORY ( %aPrincipal; )>
<!ATTLIST REPOSITORY
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED>
```

There are three kinds of render rights: PLAY, PRINT, EXPORT and VIEW. The right VIEW is included for the convenience and intuition purposes as a synonymous term of the right PLAY.

PLAY rights refer to ways of making a transient or ephemeral copy of a work available for use. For example, a page of a book may be displayed on a screen or music may be played out a speaker. The image on the screen or the sound coming out of the speaker is a transient copy. The PLAY right is also used to refer to invocation of a computer program, such as when someone plays a video game. The renderings produced when a PLAY right is exercised are ephemeral. Another way to understand a "PLAY" right is that playing requires active participation by a trusted player to render the work during the entire time that the work is perceivable.

PRINT rights refer to making more permanent rendered copies of a work outside of the control of a repository. In contrast to the Play right, once a work is printed, the printed copy can be viewed without requiring that the "PRINTER" be used the entire time. The typical case is printing something on paper at a printer. The Print right also refers to making rendered copies on other external media, such as bitmap images on removable disks or audio on magnetic tapes. For example, a repository may provide rights to "print" encrypted backup copies of a digital work. Unauthorized use of such copies is inhibited because such copies of the work are generally encrypted. An EXPORT right makes a digital source copy outside of trusted system control. Typically, this would be a file in a format suitable for unencumbered viewing, printing, or editing. Thus, this right can be used to make a digital copy that is not encrypted or otherwise protected. An EXPORT right might be exercised, for example, to release an older work after it has passed out of copyright.

Render rights can optionally specify the use of particular kinds of hardware for playing and printing. These specifications are intended to qualify the use of a work. Players and printers are parts of repositories, and are sometimes called trusted players and trusted printers. They are identified by non-transferable digital certificates. A given repository, such as a trusted printer, may have several different digital certificates corresponding to different output devices or kinds of printing services. For example, a digital movie player may specify a kind of player suitable for home use rather than theater use. The certificate for a particular player may be used to differentiate between high and low resolution renderings with appropriately different fees.

A printer specification may indicate a printer which marks the pages with a particular kind of identifying watermark. Specifications for embedding data in watermarks on trusted printers and trusted players are discussed in a later section. Here is an example of a work specification. This specification is for a digital home movie. The work has two groups of rights called “theater” and “home-viewer”. The theater rights allow playing the movie on a kind of public viewer by authorized distributors for a fee of $100. The home-viewer rights allow playing the movie on a home player for a rate of $5 per hour.

Example:

```
<?XML version="1.0" encoding="UTF-8"?>
<!DOCTYPE Work SYSTEM "xrml.dtd">
<XrML>
  <WORK>
    …
    <RIGHTSGROUP name="Theater">
        <RIGHTSLIST>
            <PLAY>
               <PLAYER><CERTIFICATE>
                 <AUTHORITY id="Motion Picture Association"></AUTHORITY>
                 <PROPERTYPAIR name="Type" value="Theater Projector"/>
               </CERTIFICATE></PLAYER>
               <ACCESS><PRINCIPAL type="User"><CERTIFICATE>
                     <AUTHORITY id="Jones Publishing"></AUTHORITY>
                     <PROPERTYPAIR name="Type" value="Distributor"/>
               </CERTIFICATE></PRINCIPAL></ACCESS>
               <FEE>
                    <MONETARY><PERUSE value="100"/><ACCOUNT>
                            <ACCOUNTTO id="Id-678-qwerqeruyt"/>
                    </ACCOUNT></MONETARY></FEE>
            </PLAY>
        </RIGHTSLIST>
    </RIGHTSGROUP>
    <RIGHTSGROUP name="Home-Viewer">
        <RIGHTSLIST>
            <PLAY>
                <PLAYER><CERTIFICATE>
                     <AUTHORITY id="Motion Picture Assoc."></AUTHORITY>
                     <PROPERTYPAIR name="Type" value="Home-Veiwer"/>
                </CERTIFICATE></PLAYER>
                <FEE>
                    <MONETARY><METEREDFEE><RATE value="5.00"/>
                        <PER hours="1"/></METEREDFEE>
                        <ACCOUNT><ACCOUNTTO id="Id-678-qwerqeruyt"/>
                        </ACCOUNT>
                   </MONETARY>
                </FEE>
           </PLAY>
        </RIGHTSLIST>
     </RIGHTSGROUP>
  </WORK>
<XrML>
```

The following example is of a digital book. The exemplar digital book has no fee for playing, but does have a five dollar fee for making a digital copy. It also has two Print rights, both requiring a trusted printer. The first Print right can be exercised if the user has a particular prepaid ticket. The second print right has a flat fee of ten dollars. The example assumes that the digital book can transmitted to a user’s computer by exercising the Copy right, and that the user can play or print the work at his or her convenience using the Play and Print rights. Fees are logged from the user’s workstation whenever a right is exercised.

Example:

```
<?XML version="1.0" encoding="UTF-8"?>
<!DOCTYPE Work SYSTEM "xrml.dtd">
<XrML>
  <WORK>
    …
   <RIGHTSGROUP name="Regular">
       <BUNDLE>
          <ACCESS><SECURITYLEVEL name="Physical" value="4"/>
          <SECURITYLEVEL name="Kernel" value="5"/></ACCESS>
          <FEE><MONETARY>
            <ACCOUNT><ACCOUNTTO id=" Account-Jones-Pub248afdoiu4398"/>
            </ACCOUNT>
          </MONETARY></FEE>
       </BUNDLE>
       <RIGHTSLIST>
        <COPY><FEE><MONETARY><PERUSE value="5"/></MONETARY></FEE></COPY>
        <TREANSFER></TREANSFER>
        <DELETE></DELETE>
        <PLAY></PLAY>
        <PRINT>
            <PRINTER><CERTIFICATE>
                <AUTHORITY id="Digital Property Trust"></AUTHORITY>
                <PROPERTYPAIR name="Class" value="Trusted Printer-6"/>
            </CERTIFICATE></PRINTER>
            <FEE><TICKET type="Prepaid-Print">
                <AUTHORITY id="Jones Publishers"></AUTHORITY>
            </TICKET></FEE>
         </PRINT>
         <PRINT>
            <PRINTER><CERTIFICATE>
               <AUTHORITY id="Digital Property Trust"></AUTHORITY>
               <PROPERTYPAIR name="Class" value="Trusted Printer-6"/>
            </CERTIFICATE></PRINTER>
            <FEE><MONETARY><PERUSE value="10"/></MONETARY></FEE>
         </PRINT>
       </RIGHTSLIST>
     </RIGHTSGROUP>
   </WORK>
</XrML>
```

Here is a different version of rights for the “Moby Dog Story” digital book. In this version, the publisher does not want digital delivery to be made directly to a consumer workstation. A practical consideration supporting this choice may be that the publisher wants to minimize the risk of unauthorized copying and requires a higher level of security than is provided by trusted systems on available workstations. Instead, the publisher wants the book to be sent directly from an on-line bookstore to a trusted printer. Printing must be prepaid via digital tickets. To enable digital distribution to authorized distributors but not directly to consumers, the publisher requires that both parties in a Copy right be have an authorizing digital license. In this example, a consumer can not play the work at a workstation. Instead, the consumer must print the work.

Example:

```
<?XML version="1.0" encoding="UTF-8"?>
<!DOCTYPE Work SYSTEM "xrml.dtd">
<XrML>
  <WORK>
    …
  <RIGHTSGROUP name="Distributor">
       <BUNDLE>
```

```
          <ACCESS><SECURITYLEVEL name="Physical" value="4"/>
            <SECURITYLEVEL name="Kernel" value="5"/></ACCESS>
          <FEE><MONETARY>
          <ACCOUNT><ACCOUNTTO id="Jones-Pub248afdoiu4398"/></ACCOUNT>
          </MONETARY></FEE>
        </BUNDLE>
        <RIGHTSLIST>
        <COPY>
          <ACCESS>
            <SOURCE><CERTIFICATE>
                 <AUTHORITY id="Jones Publishers"></AUTHORITY>
                 <PROPERTYPAIR name="Type" value="Distributor"/>
            </CERTIFICATE></SOURCE>
            <DESTINATION><CERTIFICATE>
                 <AUTHORITY id="Jones Publishers"></AUTHORITY>
                 <PROPERTYPAIR name="Type" value="Distributor"/>
            </CERTIFICATE></DESTINATION>
          </ACCESS>
          <FEE><MONETARY><PERUSE value="5"/></MONETARY></FEE>
      </COPY>
      <DELETE></DELETE>
      <PRINT>
           <PRINTER><CERTIFICATE>
                  <AUTHORITY id="Digital Property Trust"></AUTHORITY>
                  <PROPERTYPAIR name="Class" value="Trusted Printer-6"/>
           </CERTIFICATE></PRINTER>
           <FEE><TICKET type="Prepaid-Print">
                  <AUTHORITY id="Jones Publishers"></AUTHORITY>
           </TICKET></FEE>
        </PRINT>
     </RIGHTSLIST>
   </RIGHTSGROUP>
 </WORK>
<XrML>
```

4.2.3 Derivative Work Rights

Derivative work rights govern the reuse of a digital work, in whole or part, to create a new composite work. The design goal for derivative work rights is not to cover all possible forms of reuse. Many kinds of reuse involve substantial negotiation about issues that are outside the scope of what a practical trusted system can mediate or check. Rather, the derivative use rights are intended to automate the simple case where the rights owner can pre-determine fees and repository-testable conditions on a work prior to distributing it digitally.

The design goals for the derivative work rights are:

- To make it possible for the rights owner to pre-arrange fees and conditions governing subsequent use of the material in derivative works.
- To make it possible for the rights owner to pre-determine whether communication for notification or authorization is required before a derivative work right can be exercised.
- To ensure that all information about authorship, identity, contents, and rights is controlled or preserved in the derivative work.
- To arrange that the collection of fees for use of derivative works is automatic and can depend on the number of copies of the derivative work that are actually made.
- To enable a rights owner to determine whether derivative works can incorporate selected portions of a work, or whether the original work must be incorporated in total.
- To provide a framework that can eventually enable a rights owner to determine what kinds of changes are permitted in the derivative work, if any.

DTD:

```
<!ELEMENT EDIT (EDITOR?, NEXTRIGHTS?, COMMENT?, (%termsConditions;)?)>
<!ELEMENT EXTRACT (EDITOR?, NEXTRIGHTS?, COMMENT?,
                   (%termsConditions;)?)>
<!ELEMENT EMBED (EDITOR?, NEXTRIGHTS?, COMMENT?,(%termsConditions;)?)>

<!ELEMENT EDITOR ( %aPrincipal; )>
<!ATTLIST EDITOR
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED>
```

There are three kinds of derivative work rights: extract, edit, and embed.

The Extract right allows removing a portion of a digital work, creating a new work. A rights owner can divide a total digital work up into different sub-works, each with its own work specification. In this way, the rights owner can decide whether a work can be reused as a whole or in parts, and also can associate different rights and conditions with different parts of a digital work. Extract differs from Edit in that it does not grant the right to modify a work, except by removing parts of it.

The editor specification is intended to be used to test whether different editing systems qualify. From the perspective of XrML, an editor is expected to be a trusted system with signed certificates attesting to its properties. A publisher can name the properties that an editing system must have before it can be used to edit the work. If the optional editor specification is not given, then any editor program can be used for the extraction process. Otherwise, an editing system with the specified certificates can be used.

Edit grants the right to modify a work. Edit is like Extract in that it creates a new work. It differs from Extract in that it is used to make changes to a work. A rights owner can control what kinds of changes can be made to a digital work by specifying a kind of editor. For example, a particular digital audio music editor may allow someone to change the pitch or key of a piece of music, but not add notes. If the optional editor specification is not given, then any editing program can be used. Otherwise, only the specified editing system can be used.

Embed grants the right to include a work as part of a composite work. Embed puts a copy of a digital work inside a composite work. If the optional editor specification is not given, then any editing system can be used. Otherwise, only the specified editor can be used.

By default, if the optional next copy rights are not specified, the fees and conditions for a work when it is reused are the same as the fees and conditions for the original work. A publisher can override this default by adding or deleting rights in an optional NEXTRIGHTS specification.

Example:

```
<?XML version="1.0" encoding="UTF-8"?>
<!DOCTYPE Work SYSTEM "xrml.dtd">
<XrML>
  <WORK>
    …
   <RIGHTSGROUP name="Derivative Rights">
      <COMMENT>"These rights apply to deravative works."</COMMENT>
      <RIGHTSLIST>
        <COPY>
            <ACCESS><SECURITYLEVEL name="Physical" value="4"/>
              <SECURITYLEVEL name="Kernel" value="5"/></ACCESS>
            <FEE><MONETARY>
                    <PERUSE value="5"/>
                    <ACCOUNT>
                        <ACCOUNTTO id="123456789"/><HOUSE id="Visa"/>
                    </ACCOUNT>
            </MONETARY></FEE>
```

```
        </COPY>
      </RIGHTSLIST>
    </RIGHTSGROUP>
    <RIGHTSGROUP name="Consumer">
       <BUNDLE><ACCESS>
            <SECURITYLEVEL name="Physical" value="4"/>
            <SECURITYLEVEL name="Kernel" value="5"/>
       </ACCESS></BUNDLE>
       <RIGHTSLIST>
         <COPY>
            <FEE><MONETARY>
                    <PERUSE value="10.0"/>
                    <ACCOUNT>
                        <ACCOUNTTO id="123456789"/><HOUSE id="Visa"/>
                    </ACCOUNT>
            </MONETARY></FEE>
         </COPY>
         <EXTRACT>
            <EDITOR>
                  <CERTIFICATE>
                    <AUTHORITY id="DPT"></AUTHORITY>
                    <PROPERTYPAIR name="Type" value="Class 3"/>
                  </CERTIFICATE>
            </EDITOR>
         </EXTRACT>
         <EMBED>
            <EDITOR>
            <CERTIFICATE>
                    <AUTHORITY id="Adobe"></AUTHORITY>
                    <PROPERTYPAIR name="Type" value="Standard Embed"/>
                  </CERTIFICATE>
            </EDITOR>
            <NEXTRIGHTS>
                <RightsToAdd name="Derivative Rights"/>
            </NEXTRIGHTS>
         </EMBED>
         <PLAY>
            <FEE>
                <MONETARY>
                    <METEREDFEE><RATE value=".05">
                        </RATE><PER minutes="1"/></METEREDFEE>
                    <ACCOUNT>
                        <ACCOUNTTO id="123456789"/>
                        <HOUSE id="Visa"/>
                    </ACCOUNT>
                </MONETARY>
            </FEE>
         </PLAY>
         <DELETE></DELETE>
         <TREANSFER></TREANSFER>
       </RIGHTSLIST>
    </RIGHTSGROUP>
  </WORK>
<XrML>
```

Here is an example of how a work can be specified to provide for derivative use. In this example, the publisher has granted Extract and Embed rights requiring that a particular “editing” program be used. No changes can be made to the work. In addition to the original rights on the work, the embedded work has an additional copy right with a lower per use fee paid to a different account.

For some applications, it will be useful to have fairly elaborate controls on the changes that can be made to a derivative work. For example, a written work may require that certain sections always be extracted together. A musical work may allow changing the key and pitch but not the notes. The variety of possible kinds of controls is rich, not developed, and depends on the media and application. The rights language is designed to allow application-specific kinds of restrictions as described in the following.

Each kind of editor can have its own terminology for kinds of change. For example, a digital music editor might have editing operations for pitch, key, instrumentation, and other things. Digital representations for works can be augmented to include annotations that point to regions of a work and specify in an application-specific way the kinds of changes that can be made. Such annotations would have no effect when a work is played, but would control editing operations when a derived work is edited.

In this way the rights specifications are divided between domain-independent considerations in the rights markup language and the application-specific considerations in specific editors. The rights markup language allows a rights owner to specify general usage rights in the usual way, including the required use of a chosen editor (or family of editors) for editing the work. The rights owner then uses the specified editor to annotate the work to specify the kinds of changes that are allowed in domain specific terms. Later when a user wants to make a derived work by editing the work, he must use a compatible trusted editor which follows the domain-specific change annotations.

4.2.4 File Management Rights

File management rights govern access to directory and file information when two repositories are connected. File management rights control how directory information of one repository can be accessed from another. They also control the making and restoring of backup copies.

Some usage rights, such as Print or Copy, *typically* involve more than one repository. Other usage rights, such as Transfer and Loan, *inherently* involve more than one repository. To exercise rights that involve more than one repository, a user at one repository must be able to access digital works located on another.

The design goals for the file management rights are as follows:

- To provide a simple and consistent means to specify and control access to digital works across different repositories and different kinds of repositories.
- To provide a lightweight and convenient means for organizing collections of digital works on a repository.
- To integrate specifications for controlling how collections of works can be used with specifications about how individual works can be used.
- To provide facilities that manage the risk of system or media failure or distribution of rogue works.

DTD:

```
<!ELEMENT BACKUP (BACKUPRIGHTS, COMMENT?, (%termsConditions;)?)>
<!ELEMENT BACKUPRIGHTS EMPTY>
<!ATTLIST BACKUPRIGHTS   name    CDATA    #REQUIRED >
<!ELEMENT RESTORE (COMMENT?, (%termsConditions;)?)>
<!ELEMENT VERIFY (VERIFIER, COMMENT?, (%termsConditions;)?)>
<!ELEMENT FOLDER (COMMENT?, (%termsConditions;)?)>
<!ELEMENT DIRECTORY (COMMENT?, (%termsConditions;)?)>
<!ELEMENT DELETE (COMMENT?, (%termsConditions;)?)>

<!ELEMENT VERIFIER ( %aPrincipal; )>
<!ATTLIST VERIFIER
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED>
```

There are six kinds of file management rights: Folder, Directory, Delete, Verify, Backup, and Restore.

When a repository has many digital works, it is often convenient to organize them into named hierarchical sets called folders. This approach is basic to many popular file systems, such as Windows™, MacOS™, and UNIX™. In repositories built on conventional computer file systems, these hierarchical sets generally correspond to file system folders and directories. For consistency across brands of repositories, the same rights markup language that applies to digital works is also used for folders.

Folder rights govern the creation and naming of subfolders, and the right to reconfigure the folders by moving files and folders among folders. These rights are exercised by commands at a repository user interface. The right to exercise all of these operations is governed by the single "folder" right. (Restated, there are no separate rights for moving or renaming files and folders. All of these repository actions are governed under folder rights.) Directory rights govern the delivery of information about what works are contained within folders.

Here is an example of a rights group for a personal folder on a repository. The rights on a folder augment the rights of the works in a folder. For example, if a work is moved into a folder on which there are no print rights, no work in the folder can be printed. In this example, the rights group named "PersonalFolder" simply adds the requirement that a particular authorizing license ("Stefik's license") be in force before any of a set of rights can be exercised. Moving a work into this "personal" folder would prevent anyone, who is logged into the repository but who does not have the license, from accessing or using any work in the folder.

Example:

```
<RIGHTSGROUP name="PersonalFolder">
     <BUNDLE>
           <ACCESS><PRINCIPAL type="User"><CERTIFICATE>
                 <AUTHORITY id="Xerox"></AUTHORITY>
                 <PROPERTYPAIR name="ID" value="Mark Stefik"/>
           </CERTIFICATE></PRINCIPAL></ACCESS>
     </BUNDLE>
     <RIGHTSLIST>
           <PLAY></PLAY>
           <COPY></COPY>
           <LOAN></LOAN>
           <DELETE></DELETE>
           <FOLDER></FOLDER>
           <DIRECTORY></DIRECTORY>
           <TREANSFER></TREANSFER>
     </RIGHTSLIST>
</RIGHTSGROUP>
```

A directory listing may be requested of any folder in a repository, allowing either the local or a remote repository to get information about the works, their parts, and other folders contained in it. The ability to get information about a folder, a work, or its parts may be withheld in such directory listings by simply requiring a license to exercise the right.

Delete rights govern the operation of deleting a copy of a digital work. The generally expected practice is to grant any copy owner the right to delete the work. The right to delete needs to be controlled in applications where many people can log into a repository and where deleting a file could be done accidentally or in malicious mischief. To prevent the unwanted and unauthorized deletion of digital works that are accessed remotely, it is typical for a Delete right to include various conditions, such as the requirement of a digital license to exercise the right.

An opposite problem from unauthorized deletion is the creation of "Trojan Horse" works which are copied for free but which require fees to delete them. To defend against such tricks, many repositories will generate warning and confirmation messages before accepting copies of works that lack delete rights or which assess charges or conditions to exercise this right.

Verify rights are used to authorize checking of the authenticity of a work. There are several different kinds of checking that can be performed. The simplest checks verify that the work is properly encrypted. Other checks involve the use of 1-way hash functions and other keys that can detect tampering with a work. Other checks can consider hot-lists of rogue works maintained by the trusted system. Reference copies of a work may be available on the network. Local copies can be carefully compared with reference copies to verify their authenticity.

Since a verifying system has access to the contents of a digital work, publishers can require that the system be certified. To protect against "rogue verifiers" that copy works under the guise of checking them or that damage works, publishers may specify that particular kinds of verifiers be used, such as certified verifiers on well known on-line servers.

```
Example:
      <VERIFY><VERIFIER><CERTIFICATE>
            <AUTHORITY id=″DPT″></AUTHORITY>
            <PROPERTYPAIR name=″Type″ value=″Class-2″/>
      </CERTIFICATE></VERIFIER></VERIFY>
```

Backup grants the right to make copies of a work for the purpose of guarding against the loss of the original due to accident or catastrophic media or equipment failure. The backup operation creates an encrypted copy of a digital work, which is in a form where it cannot be rendered without first exercising a restore operation. The optional BackupCopyRights is used to specify the rights on the encrypted backup copy. Typically, Transfer and Restore rights would be given on a backup copy. If no backup copy rights are given, the default right is to restore the copy. Backup copy rights are not the same as the rights that will be available when the backup copy is restored. The rights on the restored copy are the same as the rights on the original before the backup was made.

A restore operation converts an encrypted backup copy into a usable copy in a controlled manner. Restore rights typically elaborate various fees and conditions that balance the interests of the copy owner with the interests of the publisher.

```
<REFERENCEDRIGHTSGROUP name="SafetyNet">
     <RIGHTSLIST>
         <TREANSFER><ACCESS>
             <SECURITYLEVEL name="Physical" value="4"/>
             <SECURITYLEVEL name="Kernel" value="5"/>
          </ACCESS></TREANSFER>
          <RESTORE>
             <ACCESS><SECURITYLEVEL name="Security-Class" value="3"/>
             </ACCESS>
             <FEE><TICKET type="Restore">
                  <AUTHORITY id="Jones-Publisher"></AUTHORITY>
             </TICKET></FEE>
          </RESTORE>
          <RESTORE>
             <ACCESS><SECURITYLEVEL name="Security-Class" value="3"/>
            </ACCESS>
            <FEE><MONETARY>
               <PERUSE value="15"/>
               <ACCOUNT><ACCOUNTTO id="Id-678-qwerqeruyt"/></ACCOUNT>
            </MONETARY></FEE>
          </RESTORE>
           <RESTORE>
                <ACCESS><SECURITYLEVEL name="Security-Class" value="3"/>
                   <PRINCIPAL><CERTIFICATE>
                        <AUTHORITY id="Murphy Publishers"></AUTHORITY>
                        <PROPERTYPAIR name="Type" value="Distributor"/>
                   </CERTIFICATE></PRINCIPAL>
                </ACCESS>
```

```
          </RESTORE>
        </RIGHTSLIST>
      </REFERENCEDRIGHTSGROUP>
      <RIGHTSGROUP name="Regular">
        <RIGHTSLIST>
            <PLAY><ACCESS>
                  <SECURITYLEVEL name="Physical" value="4"/>
                  <SECURITYLEVEL name="Kernel" value="5"/>
            </ACCESS></PLAY>
            <DELETE></DELETE>
            <TREANSFER><ACCESS>
                  <SECURITYLEVEL name="Physical" value="4"/>
                  <SECURITYLEVEL name="Kernel" value="5"/>
            </ACCESS></TREANSFER>
            <BACKUP><BACKUPRIGHTS name="Safety-Net"/></BACKUP>
        </RIGHTSLIST>
      </RIGHTSGROUP>
```

Here is an example of Backup and Restore rights. The reference rights group named “Safety-Net” includes a transfer right and three restore rights with different fees and conditions. All of the restore rights require that the repository have security class 3. One of the restore rights requires no monetary fee but uses a particular digital ticket. This approach could be used by a publisher who grants a limited number of free restorations controlled by a set of digital tickets. The second restore right costs a flat fee of $15. The third restore right is free, but requires a particular distributor’s license.

The second rights group called “regular” elaborates rights for normal use, including play, delete, transfer, and backup. The backup copy rights are those in the “SafetyNet” rights group. The next example expresses the same rights groups somewhat more simply by using Bundle specifications.

Example:

```
<REFERENCEDRRIGHTSGROUP name="SafetyNet">
     <BUNDLE>
       <ACCESS><SECURITYLEVEL name="Security-Class" value="3"/></ACCESS>
     </BUNDLE>
     <RIGHTSLIST>
       <TREANSFER><ACCESS>
          <SECURITYLEVEL name="Physical" value="4"/>
          <SECURITYLEVEL name="Kernel" value="5"/>
       </ACCESS></TREANSFER>
       <RESTORE>
          <FEE><TICKET type="Restore">
               <AUTHORITY id="Jones-Publisher"></AUTHORITY>
          </TICKET></FEE>
       </RESTORE>
      <RESTORE>
          <FEE><MONETARY>
               <PERUSE value="15"/>
               <ACCOUNT><ACCOUNTTO id="Id-678-qwerqeruyt"/></ACCOUNT>
          </MONETARY></FEE>
      </RESTORE>
      <RESTORE>
          <ACCESS><PRINCIPAL><CERTIFICATE>
                <AUTHORITY id="Murphy Publishers"></AUTHORITY>
                <PROPERTYPAIR name="Type" value="Distributor"/>
          </CERTIFICATE></PRINCIPAL></ACCESS>
       </RESTORE>
    </RIGHTSLIST>
</REFERENCEDRIGHTSGROUP>
<RIGHTSGROUP name="Regular">
```

```
    <BUNDLE><ACCESS>
        <SECURITYLEVEL name="Physical" value="4"/>
        <SECURITYLEVEL name="Kernel" value="5"/>
    </ACCESS></BUNDLE>
    <RIGHTSLIST>
            <PLAY></PLAY>
            <DELETE></DELETE>
            <TREANSFER></TREANSFER>
            <BACKUP><BACKUPRIGHTS name="Safety-Net"/></BACKUP>
      </RIGHTSLIST>
</RIGHTSGROUP>
```

4.2.5 Configuration Rights

Configuration rights govern the adding and removing of system software from highly secure repositories. These rights are not relevant on platforms where the loading of software is not controllable.

The design goals for configuration rights are as follows:

- To provide a means to load software that insures that it is certified, has not been tampered with, and is compatible with the repository.
- To arrange that a user installing and uninstalling repository software in the field has security on the same order as when software is installed in a secure facility.

Because general purpose computers have other means for loading and deleting software, configuration rights are most relevant to dedicated, high security repositories. Typically, the control offered by configuration rights on secure repositories is most useful when adding software for a new player or editor.

```
DTD:

<!ELEMENT INSTALL (COMMENT?, (%termsConditions;)?)>
<!ELEMENT UNINSTALL (COMMENT?, (%termsConditions;)?)>
```

There are two kinds of configuration rights: Install and Uninstall. As with all other rights, these rights also have an optional comment, time, access, and fee specification. An Install operation makes software runnable on a repository. Just copying a program to a repository does not make it runnable. The installation operation checks that software is certified, that it has not been tampered with, and that it is compatible with the repository. If these conditions are satisfied, the install operation links the software into the secure software procedures of the repository.

The Uninstall operation removes software from the running system. Uninstall does not delete the file corresponding to the program. It merely disables it from running, restoring it to the state before it was installed.

4.3 Specifying Times

Usage rights are usually granted for a specified period of time. The time period over which a right is granted is indicated by a time specification. To honor time specifications, a trusted system must have accurate and tamper-proof clocks. The language design goals of the time specifications are:

- To express time limits that govern when rights become available and when they cease to be available.
- To express the categories of time blocks that are useful for governing commercial use of digital works.
- To describe contiguous periods of time, periods of time that do not start until a work is first used, and accumulative amounts of time that need not be contiguous.
- To provide unambiguous specifications suitable for international commerce and commerce across time zones.

Time specifications use representations of moments in time and units of time. The next two sections describe representations for moments and time units. We then discuss time specifications for usage rights that use these representations to indicate when the rights are valid.

4.3.1 Specifying Moments in Time

Moment specifications indicate a particular moment in time. All specifications of a particular moment in time must specify the date and optionally the time.

```
DTD:

<!ENTITY % moment "#PCDATA" >
```

The format used in this document for representing a time moment is "yyyy-mm-ddThh:mm:ss"; For instance, 2000-01-27T15:30 specifies Three-thirty p.m. on January Twenty-Seventh, year Two Thousand. Time constants, hh:mm:ss, represent specific times-of-day on the 24-hour clock in Universal Time, or UT (equivalent to Greenwich Mean Time, or GMT). The "hh" represents the hour and therefore must be between *0* and *23*, inclusive. The "mm" represents the minute and must be between *0* and *59*, and the "ss" represents the second, and therefore must be between 0 and 59, inclusive. The time "24:00:00", in some contexts defined to be equivalent to the time "00:00:00", is **not** allowed.

Example:

Time of Day	**Meaning**
17:30:01	Five-thirty p.m. and one second. (GMT)
00:00:00	Twelve o'clock Am. (GMT) (midnight, the first second of the day)
12:00:00	Twelve o'clock p.m. (noon, GMT)

Date constants, yyyy-mm-dd, represent a specific day on the calendar. "yyyy" represents the year and must be between 0 and 9999, inclusive. Note that one-, two-, and three-digit integers represent years through 999 AD. Specifically, two-digit integers **do not**, as in common usage, represent years in the current centennial. "mm" represents the month and therefore must be between 1 and 12, inclusive. "dd" represents the day and therefore must be between 1 and either 28, 29, 30, or 31, inclusive, according to the calendar. Here are some examples of date constants.

Example:

Date Constant	**Meaning**
1996-5-15	May 15, 1996
0999-12-31	December 31, 999
2000-01-03	January 3, 2000
2000-01-04	

Since repositories need to have synchronized clocks for their transactions, which may take place across time zones, it is convenient to use Universal Time as a standard. Conversion to local date and time is performed by the user interface of the trusted system.

4.3.2 Specifying Units of Time

Time specifications indicate a period of time by specifying a number of calendar units, a number of time units, or both. Calendar units are specified in calendar years, months, and days. Time units are specified in hours, minutes, and seconds.

Calendar units are idiosyncratic as units of measure because they do not have uniform sizes. Thus, years have different numbers of days in them depending on leap years. Similarly, different months have different numbers of days in the Julian calendar. Nonetheless, these "units" are convenient in specifications because so many people use calendars.

```
DTD:

<!ENTITY % interval
```

```
"years        CDATA    #IMPLIED
 months       CDATA    #IMPLIED
 days         CDATA    #IMPLIED
 hours        CDATA    #IMPLIED
 minutes      CDATA    #IMPLIED
 seconds      CDATA    #IMPLIED">
```

Calendar units (years, months, days) have the same syntactic structure as date constants. The number of years can be between 0 and 9999, referring to a number of calendar years. The number of months can be between 0 and 9999, referring to a number of calendar months. The number of days can be between 0 and 9999. Here are some examples of calendar units.

Example:

Calendar Units	**Meaning**
months=3 days=4	Three months and four days.
days=40	Forty days.
days=7	Seven days
years=1	One calendar year.

In particular, a specification 00/03/00 for three months is not the same as 00/00/90 for ninety days, although they are close. The actual number of days in three calendar months depends on the months when the specification applies.

Time units (hours, minutes, seconds) have the same syntactic structure as time constants. The number of seconds can be an integer between 0 and 9999. The number of minutes can be an integer between 0 and 9999. The number of hours can be an integer between 0 and 9999.

Example:

Time Units	**Meaning**
hours=2 minutes=30	Two hours and thirty minutes.
minutes=67	Sixty seven minutes.
minutes=5 seconds=30	Five and a half minutes.

4.3.3 Specifying When Rights Can Be Exercised

We are now ready to consider time specifications for rights. All time specifications refer to a time interval over which the rights are enabled. In addition, there is an optional expiration date, after which the right cannot be exercised even if the interval is not exhausted. If no time specification is given, the right is assumed to be always valid.

DTD:

```
<!ELEMENT TIME ( (FROM | INTERVAL | METERED)?, UNTIL)>
<!ELEMENT FROM ( %moment; )>
<ELEMENT INTERVAL EMPTY>
<!ATTLIST INTERVAL %interval;>
<!ELEMENT METERED EMPTY>
<!ATTLIST METERED %interval;>
<!ELEMENT UNTIL ( %moment; )>
```

There are three kinds of intervals in time specifications: fixed intervals, sliding intervals, and metered intervals.

In their most complete form, fixed intervals start at a fixed time and ends at a fixed time. In variations, a fixed interval specification can specify only a starting time or only an ending time. Here are some examples of fixed interval specifications.

Examples:

- `<TIME><Until>1995-01-01</UNTIL></TIME>`

- <TIME><Until>1994-12-31T23:59:59</UNTIL></TIME>
  Can be used until midnight New Year's Eve in 1994.

- <TIME>
    <FROM>1997-02-14</FROM>
    <UNTIL>1997-02-15</UNTIL>
  </TIME>
  Can be used only on Valentine's Day in 1997.

Sliding intervals specify a consecutive period of time. The period starts when the right is first exercised. To accurately account for sliding intervals, a trusted system must keep track of the time that use started. Here are some examples of sliding interval specifications.

Examples:

- <TIME><INTERVAL hours="12"/><UNTIL>1996-01-01</UNTIL></TIME>
  Can be used for any twelve consecutive hours any time up to January 1, 1996.

- <TIME><INTERVAL hours="2"/><UNTIL>1996-05-01T8</UNTIL></TIME>
  Can be used for two consecutive hours any time up to 8:00 am May 1, 1996.

Metered intervals represent accumulative periods of time. A user can start and stop using exercising a right as often as he wants. The metering clock runs whenever the right is being exercised, and stops when the user stops using it. The right can be exercised as long as the total time has not been used. To accurately account for metered time, a repository must keep an incremental log of the amount of time used so far.

Examples:

- <TIME>
    <METERED hours="2"/><UNTIL>1996-05-01T8</UNTIL>
  </TIME>
  Can be used for two hours of metered time spread out in any way up to 8:00 am on May 1, 1996. The work cannot be used after May 1, 1996 even if the metered time has not yet exhausted the two hours.

- <TIME>
    <METERED months="2"/><UNTIL>1998-09-01T8</UNTIL>
  </TIME>
  Can be used for two months spread out in any way until September 1, 1998.

Note that a metered time interval specification is not the same thing as a metered fee specification. A metered time interval indicates the time that a right is valid. It can be used even on rights for which there is no fee. A metered fee specification is a way of indicating *how to charge* to exercise a right in using a work. Metered fee specifications could be used even on works where the right to use the work does not expire at all.

4.4 Specifying Fees and Incentives

Usage rights can have associated fees which are paid to specific accounts. To honor these fees, trusted systems need to communicate regularly with financial clearing houses. They also need to have reliable and tamper-proof means of recording and reporting billing data. Different repositories have different approaches, depending on the application and the required levels of security and accountability. Some systems accumulate billing logs locally and report them to financial clearing houses on a scheduled basis. Less secure systems must connect with an on-line financial clearinghouse each time that a transaction is made, and before a transaction can be completed.

The language design goals for fee specifications are:

- To express the kinds of charges to exercise a right that are commercially useful for all kinds of digital works.
- To express variations on per-use fees and metered-use fees, including payments per time unit, minimum and maximum fees over a period, and fixed fees for unlimited use in a fixed time period.
- To accommodate approaches based on both credit accounts and debit accounts.
- To support transactions on trusted systems where the consumer is not on-line.
- To express fees in most national currencies and also in publisher-supplied non-currency units (digital tickets).
- To express payments from a consumer to a publisher as well as incentives from a publisher to a consumer.
- To express predetermined and scheduled changes in price.
- To express bundled prices for using combinations of rights.

A fee specification describes the financial transaction required to exercise a right. It can include a ticket specification or a monetary specification. If a ticket specification is given, then the transaction requires that a particular kind of digital ticket be processed ("punched") by a digital ticket agent accessible to the repository. The simplest digital tickets are used just once and are punched by a standard digital ticket agent when they are used. More complex tickets have more internal structure, representing expiration dates, numbers of times that they can be punched, and may require a specialized ticket agent. If a monetary specification is given, then the user will be billed.

DTD:

```
<!ELEMENT FEE ( TICKET | MONETARY )>
<!ELEMENT TICKET (AUTHORITY)>
<!ATTLIST TICKET
  type     CDATA    #IMPLIED
  id       CDATA    #IMPLIED >
<!ELEMENT AUTHORITY EMPTY>
<!ATTLIST AUTHORITY
  type     CDATA    #IMPLIED
  id       CDATA    #IMPLIED >
```

Each right of a digital work can have different fee specifications. In the following example, the "Play Options" rights group gives three different fee schemes for playing the work. One has a ticket, one charges a dollar per play, and one is fifty cents an hour. Some usage rights in the "Regular" rights group, such as transfer, delete, and backup require no fees at all.

Example:

```
<?XML version="1.0" encoding="UTF-8"?>
<!DOCTYPE Work SYSTEM "xrml.dtd">
<XrML>
  <WORK>
    …
    <CONTENTS from="1" to="16636"/>
    <RIGHTSGROUP name="Play Options">
      <RIGHTSLIST>
            <PLAY><FEE><TICKET type="Demo">
                    <AUTHORITY id="Jones Publishing"></AUTHORITY>
            </TICKET></FEE></PLAY>
            <PLAY>
               <TIME>
                   <FROM>1997-02-14</FROM><UNTIL>1998-02-15</UNTIL>
               </TIME>
               <FEE><MONETARY>
                    <PERUSE value="1.0"></PERUSE>
                    <ACCOUNT><ACCOUNTTO id="Id-677-qp3498"/></ACCOUNT>
```

```
              </MONETARY></FEE>
            </PLAY>
            <PLAY>
               <TIME><UNTIL>2000-01-01</UNTIL></TIME>
               <FEE><MONETARY>
                    <METEREDFEE><RATE value=".50"></RATE></METEREDFEE>
                    <ACCOUNT><ACCOUNTTO id="678-qwerqeruyt"/></ACCOUNT>
              </MONETARY></FEE>
          </PLAY>
      </RIGHTSLIST>
  </RIGHTSGROUP>
  <RIGHTSGROUP name="Regular">
      <RIGHTSLIST>
          <TREANSFER>
                <ACCESS>
                      <SECURITYLEVEL name="Physical" value="4"/>
                      <SECURITYLEVEL name="Kernel" value="5"/>
                </ACCESS>
          </TREANSFER>
          <COPY>
                <ACCESS>
                      <SECURITYLEVEL name="Security-Class" value="3"/>
                </ACCESS>
                <FEE><MONETARY>
                      <PERUSE value="1.0"></PERUSE>
                      <ACCOUNT>
                      <ACCOUNTTO id="Id-678-qwerqeruyt"/></ACCOUNT>
                </MONETARY></FEE>
          </COPY>
          <DELETE></DELETE>
          <BACKUP></BACKUP>
          <RESTORE>
               <ACCESS>
                      <SECURITYLEVEL name="Physical" value="4"/>
                      <SECURITYLEVEL name="Kernel" value="5"/>
               </ACCESS>
               <FEE><MONETARY>
                 <PERUSE value="5.0"/>
                 <ACCOUNT><ACCOUNTTO id="678-qwerqeruyt"/></ACCOUNT>
               </MONETARY></FEE>
          </RESTORE>
      </RIGHTSLIST>
   </RIGHTSGROUP>
 </WORK>
</XrML>
```

More details on ways to specify different kinds of fees are described in the following sections.

4.4.1 Currencies and Accounts

Charges may be expressed in any currency known to the repository. As with paper currency, digital bills are subject to the principles and dynamics of currency exchange when accounts are reconciled with financial clearing houses. Restated, the transactions are recorded in a particular currency. Subsequent fluctuations in the value of the currency may occur. If the currency is not explicitly specified, then “US dollar” is assumed. A table of ISO currency codes is given in Appendix A.

DTD:

```
<!ELEMENT CURRENCY EMPTY>
<!ATTLIST CURRENCY iso-code      CDATA    #REQUIRED>
<!ELEMENT ACCOUNT ( ( ACCOUNTTO, HOUSE? ) | ( ACCOUNTFROM, HOUSE? ) )>
```

```
<!ELEMENT ACCOUNTTO EMPTY>
<!ATTLIST ACCOUNTTO
  id            CDATA    #REQUIRED
  type          CDATA    #IMPLIED
  url           CDATA    #IMPLIED>
<!ELEMENT ACCOUNTFROM EMPTY>
<!ATTLIST ACCOUNTFROM
  id            CDATA    #REQUIRED
  type          CDATA    #IMPLIED
  url           CDATA    #IMPLIED>
<!ELEMENT HOUSE EMPTY>
<!ATTLIST HOUSE
  id            CDATA    #REQUIRED
  type          CDATA    #IMPLIED
  url           CDATA    #IMPLIED>
```

Usage rights fees and incentives may be handled by a wide variety of financial systems and institutions including banks, credit institutions, and electronic commerce organizations.

The account specification typically indicates the account to be paid. Fees are paid by the user/copy-owner to the revenue-owner and indicated by the element `ACCOUNTTO` in the account specification. Incentives are paid by the revenue-owner to user/copy-owner and are indicated by the element `ACCOUNTFROM` in the account specification.

An `id` in `ACCOUNTTO` and `ACCOUNTFROM` is an account number together with an institution code that identifies a specific account at a specific bank or other financial institution from which fees will be collected, or to which incentives will be paid.

The account specification takes an optional clearinghouse specification that names a financial clearing house. In closed systems where the financial clearing house is known by default, the specification is unnecessary. In open systems where multiple clearinghouses are possible and account numbers are not unique, the specification names the clearinghouse to be used. In the following examples, exercising the first right causes the user to be billed fifty cents. Exercising the second one causes the user to be credited fifty cents. The right itself includes information known to the publisher. It does not, for example, name the user's account. The user's account information must be held elsewhere in the system.

`Example:`

```
<PLAY>
      <FEE><MONETARY>
            <PERUSE value=".50"></PERUSE>
            <ACCOUNT><ACCOUNTTO id= "245lkj5987"/></ACCOUNT>
      </MONETARY></FEE>
</PLAY>
<PLAY>
      <FEE><MONETARY>
            <PERUSE value=".50"></PERUSE>
            <ACCOUNT><AccountFrom id= "245lkj5987"/></ACCOUNT>
      </MONETARY></FEE>
</PLAY>
<PLAY>
      <FEE><MONETARY>
            <PERUSE value=".50"></PERUSE>
            <ACCOUNT>
                  <ACCOUNTTO id= "245lkj5987"/>
                  <HOUSE id="Visa"/>
            </ACCOUNT>
      </MONETARY></FEE>
</PLAY>
```

4.4.2 Digital Tickets

Digital tickets are used for pre-paid uses. Loosely speaking, they are a kind of private currency. They are not legal tender, but they can often be purchased for money. They can be exchanged for particular services. Like movie tickets, digital tickets are punched when they are used. On a trusted system, a digital ticket is punched by a local digital ticket agent. A ticket is a digital certificate that is altered ("punched") after use making it unusable for any future transactions.

The design goals for digital tickets are:

- Digital tickets are special digital works issued by publishers.
- A ticket publisher can predetermine the conditions under which tickets can be created, transferred across repositories, or used in rights.
- Independent of the time specifications on rights that use them, tickets can expire even if they have not been used.
- Both unused and punched tickets are secure against forgery and tampering.
- Tickets are processed by a generic ticket agent that is part of a repository.
- Tickets can specify additional required processing by local or remote ticket agents.
- Metered tickets can be used to prepay usage over a period of time.
- Metered tickets can be shared across multiple rights, so that the rights share a single clock.

Digital tickets are implemented as encrypted computer files kept in trusted storage. Repositories treat digital tickets much like digital works, in that digital tickets have rights attached to them and these rights are used to govern copying, transferring, and other uses. A digital ticket is punched when a digital ticket agent "plays" the ticket. In the typical and simplest case, playing a digital ticket involves recording its use in the billing data and marking it as punched.

More sophisticated kind of digital tickets are possible. For example, some digital tickets can be "metered tickets" which keep track of the amount of time over which they are used. In this case, each time that the ticket is used, the digital ticket agent records on the ticket the time that passes when a right is being exercised. The ticket is exhausted when the time runs out. More than one right can be associated with a given metered ticket. In this case, the multiple rights share a clock. Metered tickets are also useful when the time that rights will be used is paid ahead of time, instead of being logged while the right is exercised. The internal state of tickets, including metering information, is part of the ticket creation process and not part of the language for describing rights that use such tickets. See the section on tickets and digital certificates.

If a ticket has Copy and Transfer rights, copies can be made (possibly for a fee) for another user. If a ticket has been used before it is copied, it must be refreshed before it is copied.

Digital tickets are accessible only to the certified accounting routines of the trusted system. Because they are tokens for payment, tickets must be kept in trusted storage on trusted and semi-trusted systems. For certain low-security repositories, tickets can be accessed from remote repositories of adequate security. Software-only trusted systems can still use digital tickets, but their security is necessarily less.

Example:

```
<PLAY>
      <TIME><UNTIL>2050-01-01</UNTIL></TIME>
      <FEE><TICKET type="Introductory Offer">
            <AUTHORITY id=”Jones Publishers”></AUTHORITY>
      </TICKET></FEE>
</PLAY>
<PLAY>
      <TIME><UNTIL>2010-01-01</UNTIL></TIME>
      <FEE><MONETARY>
            <METEREDFEE>
                  <RATE value=”1.0”></RATE>
```

```
            <PER hours=”1”/>
        </METEREDFEE>
        <ACCOUNT>
            <ACCOUNTTO id=”Jones-Pub-poierlkj45”/>
            <HOUSE id=”CitiBank”/>
        </ACCOUNT>
    </MONETARY></FEE>
</PLAY>
```

In this example, there are two play rights. The first one just requires a ticket. The second example requires no ticket, but costs a dollar per hour.

4.4.3 Per Use and Metered Fees

Regular fee specifications are the basic forms of monetary arrangements. For each fee, specifications can include a fee type, optional minimum and maximum price specifications over the life of this copy of the work, and the account to be charged (or credited to the user as an incentive).

DTD:

```
<!ELEMENT MONETARY ( ( PERUSE | METEREDFEE | BESTPRICEUNDER |
                       CALLFORPRICE | MARKUP),
                     MIN?, MAX?, ACCOUNT )>
<!ELEMENT PERUSE (CURRENCY?)>
<!ATTLIST PERUSE value          CDATA    #REQUIRED>
<!ELEMENT METEREDFEE (RATE, PER, BY?)>
<!ELEMENT RATE (CURRENCY?)>
<!ATTLIST RATE value          CDATA    #REQUIRED>
<!ELEMENT PER EMPTY>
<!ATTLIST PER %interval;>
<!ELEMENT BY EMPTY>
<!ATTLIST BY %interval;>
<!ELEMENT BESTPRICEUNDER (CURRENCY?)>
<!ATTLIST BESTPRICEUNDER value          CDATA    #REQUIRED>
<!ELEMENT CALLFORPRICE EMPTY>
<!ATTLIST CALLFORPRICE dealer-id     CDATA    #REQUIRED>
<!ELEMENT MIN (RATE, PER)>
<!ELEMENT MAX (RATE, PER)>
```

The optional minimum price specification indicates the minimum price to be paid per time unit if the right is exercised at all. The optional maximum price specification refers to the maximum price to be paid per time unit if the right is exercised at all. When both a maximum and minimum price specification are given, the maximum price specification dominates.

A Per-Use fee is a simple fee charged every time that a right is exercised. Here are some examples of per-use fees.

Example:

```
 <PLAY>
     <FEE><MONETARY>
          <PERUSE value=”.50”></PERUSE>
          <ACCOUNT>
               <ACCOUNTTO id= “Account Jones-Pub-245lkj5987”/>
               <HOUSE id=” American Express”/>
          </ACCOUNT>
     </MONETARY></FEE>
 </PLAY>
 <PLAY>
     <FEE><MONETARY>
```

```
            <PERUSE value=”15”></PERUSE>
            <MAX><RATE value=”15.00”>”USD”</RATE><PER ></Per></MAX>
            <ACCOUNT>
                  <ACCOUNTTO id= “Account-Jones-Pub248afdoiu4398”/>
                  <HOUSE id=” MasterCard”/>
            </ACCOUNT>
      </MONETARY></FEE>
</PLAY>
```

In this example, the user has three different per use options for playing the digital work. To exercise the first right, the cost is simply fifty cents per use. To exercise the second right, the user pays $15 the first time he plays the work and all subsequent plays are free. This is arranged in the language by specifying a $15.00 per use fee together with a lifetime maximum fee of $15. This kind of billing arrangement can be used with consumer-based distribution schemes based on pay-for-play rather than pay-for-copy.

MeteredFee specify a fee charged by the hour, or more generally, a fee charged per given units of time in use. Metered fees only make sense for uses whose value depends on time. Thus, the *MeteredFee* specification is only valid in conjunction with the Play right.

The *Per* portion of the specification indicates the units of time over which the rate is charged. The optional *By* field indicates the resolution at which billing is computed, that is, whether it is computed to the nearest day, minute, or second. If the *By* element is not given, it is assumed to be the same as the *Per* field.

For example, a per year charge is figured over a calendar year, starting at the time the work is used. For example, suppose that a user is using a work charged at $365 per year. He starts on December 31, 1996 and ends on January 1, 1997. Assume that the duration of use is approximately one day and that the *By* field is one day or less. With these assumptions, the fee for use would be about one dollar. If the By: field is set to a year, then the fee would be about $365, although the user would not be charged more for exercising the right further during 1997.

```
Example:

<PLAY>
      <TIME><UNTIL>2050-01-01</UNTIL></TIME>
      <FEE><MONETARY>
            <METEREDFEE><RATE value=”.50”></RATE>
                  <PER hours=”1”/></METEREDFEE>
            <ACCOUNT><ACCOUNTTO id= “Account Jones-Pub-245lkj5987”/>
      </ACCOUNT>
      </MONETARY></FEE>
</PLAY>
<PLAY>
      <TIME><UNTIL>2050-01-01</UNTIL></TIME>
      <FEE><MONETARY>
            <METEREDFEE>
                  <RATE value=”.50”></RATE><PER hours=”1”/>
                  <BY seconds=”1”/></METEREDFEE>
            <ACCOUNT><ACCOUNTTO id= “Account Jones-Pub-245lkj5987”/>
      </ACCOUNT>
      </MONETARY></FEE>
</PLAY>
```

Here are two examples of metering both with a rate of fifty cents per hour. Suppose that someone plays a work for 70 minutes. In the first example, the user is charged $1. The metering counts how many of the intervals (“two 1-hour ticks”) are started. As soon as the user plays the work past the first hour, he is charged for two hours. In the second example, the *By* field is specified to be one second. In this case, the user is only charged for the portion of the second hour that he uses computed to the nearest second. Rounding up to the nearest second, he would be charged about $.59 for using the work for 70 minutes.

The following example combines a metered rate with a minimum fee. To exercise the right, the user is charged a dollar per hour with a minimum fee of $2 per day but a maximum of $15 per year.

Example:

```
<PLAY>
     <TIME><UNTIL>2050-01-01</UNTIL></TIME>
     <FEE><MONETARY>
           <METEREDFEE><RATE value=”1.00”></RATE>
                 <PER minutes=”1”/></METEREDFEE>
           <MIN><RATE value=”2.00”></RATE><PER minutes=”1”/></MIN>
           <MAX><RATE value=”15.00”></RATE><PER hours=”1”/></MAX>
           <ACCOUNT>
                 <ACCOUNTTO id= “345678”/>
                 <HOUSE id=”Great Western”/>
           </ACCOUNT>
     </MONETARY></FEE>
</PLAY>
```

Another useful billing variation is to have a fixed fee for unlimited use of a work over a fixed period of time. Restated, the user should be able to exercise the right as many times as desired during a fixed period for an overall fixed fee. This arrangement can be specified by a combination of time specifications and fee specifications. The example shows two ways to grant the right to play a work as often as desired during any contiguous month any time up to 2010. The *Interval* time specification says that the month starts when the work is first played. The *MeteredFee* specification and *PerUse* specifications establish the rate. In both examples, the *Max* specification assures that the user will not be charged more than $10 during the month for exercising the right.

Example:

```
<PLAY>
     <TIME><INTERVAL months=”1”/><UNTIL>2010-01-01</UNTIL></TIME>
     <FEE><MONETARY>
           <METEREDFEE><RATE value=”10”></RATE>
                 <PER months=”1”/></METEREDFEE>
           <MAX><RATE value=”10”></RATE><PER months=”1”/></MAX>
           <ACCOUNT><ACCOUNTTO id= “Jones-Pub-245lkj5987”/></ACCOUNT>
     </MONETARY></FEE>
</PLAY>
<PLAY>
     <TIME><INTERVAL months=”1”/><UNTIL>2010-01-01</UNTIL></TIME>
     <FEE><MONETARY>
           <PERUSE value=”10”></PERUSE>
           <MAX><RATE value=”10”></RATE><PER months=”1”/></MAX>
           <ACCOUNT><ACCOUNTTO id= “Jones-Pub-245lkj5987”/></ACCOUNT>
     </MONETARY></FEE>
</PLAY>
```

4.4.4 Best-Price Fees

Best-Price is a fee that can be dynamic and is determined when the account is settled. It is used to accommodate special deals, rebates, and pricing that depends on information that is not available to the trusted repository at the time the usage right is exercised, but without communicating with a dealer before the purchase is authorized. A Best-Price specification limits the risk to the user by naming a maximum amount that the exercising of the right will cost. This is the amount that is tentatively charged to the account. However, when the transaction is ultimately reconciled, any excess amount charged will be returned to the user/copy-owner in a separate transaction. In the example, the is charged $5 to play the work, but will get a rebate if the price has gone down.

Example:

```
<PLAY>
     <TIME><UNTIL>2050-01-01</UNTIL></TIME>
     <FEE><MONETARY>
           <BestPriceUnder value=”5”/>
           <ACCOUNT><ACCOUNTTO id=”Jones-Pub-245lkj5987”/></ACCOUNT>
     </MONETARY></FEE>
</PLAY>
```

Call-For-Price is similar to Best-Price in that it is intended to accommodate cases where prices are dynamic. However, unlike Best-Price, communication with a dealer to determine the price is required before the purchase is authorized; the transaction cannot be completed if the trusted repository is unable to communicate with the dealer.

Example:

```
<PLAY>
     <FEE><MONETARY>
           <CALLFORPRICE dealer-id=”Jones-Network-Sales-lkjqerk8743h”/>
           <ACCOUNT><ACCOUNTTO id=”Jones-Pub-245lkj5987”/></ACCOUNT>
     </MONETARY></FEE>
</PLAY>
```

4.4.5 Markup Fees

Markup fees are fees that are computed as a percentage of other fees. For example, a distributor may want to add a flat ten percent overhead for selling copies of a digital work, or a government may want to tax sales of a digital works.

DTD:

```
<!ELEMENT MARKUP EMPTY>
<!ATTLIST MARKUP percentage    CDATA    #REQUIRED>
```

Markup fees are specified in conjunction with composite works. In the following example, the Royal Tax Collector of Oz has created a work specification, which can be added to the specifications of all digital works sold in Oz. This specification adds a one percent tax every time a copy right is exercised in the embedded digital work. It does not tax the exercising of any other right.

Example:

```
<?XML version="1.0" encoding="UTF-8"?>
<!DOCTYPE Work SYSTEM "xrml.dtd">
<XrML>
  <WORK>
    …
    <RIGHTSGROUP name="Oz Taxes ">
        <RIGHTSLIST>
            <COPY>
            <FEE>
                  <MONETARY><Markup percentage=".01"/></MONETARY>
                    <MIN><RATE value=".05"/><PER seconds="1"/></MIN>
                  <ACCOUNT>
                    <ACCOUNTTO id="Royal-Treasury-of-Oz-193487fi43987"/>
                  </ACCOUNT>
            </FEE>
            </COPY>
            <TREANSFER></TREANSFER>
            <LOAN></LOAN>
            <PLAY></PLAY>
```

```
            <PRINT></PRINT>
                <DELETE></DELETE>
            <Directory ></DIRECTORY>
                <FOLDER></FOLDER>
               <BACKUP></BACKUP>
               <RESTORE></RESTORE>
        </RIGHTSLIST>
      </RIGHTSGROUP>
  </WORK>
</XrML>
```

The minimum fee specification within the Markup specification assures that a minimum tax of five cents will be collected to make copies even if the work has no publisher-set fee for copying.

4.5 Specifying Access Controls

Although one of the main advantages of digital works is the potential for broad and inexpensive distribution, it is sometimes important to limit the distribution of particular digital works. For example, a company using digital documents to coordinate distributed work teams may want to limit distribution to certain employees in order to protect corporate or trade secrets. A publisher of an electronic journal may want to assure that copies of a work are only available to subscribers or subscribing institutions. A public school may want to limit access by students to documents that are age appropriate or which reflect certain community values. In these examples, limiting distribution is different from making sure that people have paid for works that they use. Access specifications for authorization enable non-monetary criteria to govern the exercise of usage rights.

In addition to the requirement to authorize distribution of works to particular sets of people, there is a requirement to limit distribution of works to particular repositories. Repositories will be created by different vendors for different purposes and will have different kinds of security arrangements. Creators of digital works need to be able to specify the required levels of security. When two repositories establish a communications link, part of the process is to mutually determine levels of security.
The approach for both authorization and security is the controlled distribution of digital certificates or digital licenses. A digital license is analogous to a driver's license in that the right to drive a car is supposed to be governed by whether a person is qualified. The operational test that a person has passed a driver's test is possession of a license. The operational test that a repository or owner is qualified is roughly possession of a license, where the license is held in secure storage. A preferred variation on simple "possession of the license" is to have a challenge/response protocol in which the user's system proves that it is the bonafide holder of the license, based on the use of their private key. (The particulars of the protocol are outside the scope of this document.) Digital licenses are themselves digital works, have provisions for detecting tampering, and may be held in trusted storage on repositories.

The rights markup language addresses security concerns by recognizing that different digital platforms can have different levels of security. When two repositories communicate, the first step of their registration process includes a challenge/response protocol during which they determine each other's level of security. The level of security of a repository is established by a certification process, which assigns a repository a non-transferable digital certificate indicating its security level.

The design goals for authorization specifications in the rights markup language are:

- It should be possible to specify different authorization requirements for different rights on a work.
- Digital licenses should be issuable potentially by any registered organization. It should be possible to determine the issuer from the license itself.
- Digital licenses for repository operators should be issuable to persons and should become visible to transactions when the authorized person logs onto the repository.
- Digital licenses for repositories should be issuable to repositories by certifying organizations. These licenses are accessible when anyone uses the repository.
- Digital licenses should have expiration dates.

- Distribution of digital licenses should be reliably controllable. A license issuer can predetermine the conditions under which licenses can be created or transferred across repositories.
- For transactions involving two repositories, it should be possible to specify different authorization requirements for each repository.
- It should be possible to specify authorizations which require communication or computation in order to complete a transaction.

In the rights markup language, requirements for authorization are indicated by Access: specifications. The particular design goals for security class specifications are given later.

```
DTD:

<!ELEMENT ACCESS (PRINCIPAL*, SECURITYLEVEL*, SOURCE?, DESTINATION?)>
<!ELEMENT SECURITYLEVEL EMPTY>
<!ATTLIST SECURITYLEVEL
  name          CDATA    #IMPLIED
  value         CDATA    #REQUIRED>
<!ELEMENT SOURCE ( %aPrincipal; )>
<!ATTLIST SOURCE
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED>
<!ELEMENT DESTINATION ( %aPrincipal; )>
<!ATTLIST DESTINATION
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED>
<!ELEMENT CERTIFICATE (AUTHORITY, PROPERTYPAIR*)>
<!ELEMENT PROPERTYPAIR EMPTY>
<!ATTLIST PROPERTYPAIR
  name          CDATA    #REQUIRED
  value         CDATA    #REQUIRED>
```

Access specifications include principal authorization specifications, security class specifications, source authorization specifications, and destination authorization specifications.

4.5.1 Digital Licenses (Certificates)

We use the terms authorizations, certificates, and digital licenses interchangeably to refer to digital objects, signed by a known digital authority, that attest that certain persons or repositories have certain properties. For example, an identity certificate issued to a person might contain certain identifying address, employment, age, club membership, or citizenship information, as well as a public key issued to the person. A certificate issued to a repository might contain information about its serial number, its country of registration, it security classification, its registered owner.

User authorization specifications refer to authorizations associated with a person requesting a transaction on a repository. A repository must be able access separate sets of authorizations associated with each person who uses it. In some systems, this separation of authorizations would be accomplished by having the authorizations kept on a repository card that a user plugs into the file repository in order to use it. In other implementations, the certificates for different people may be kept on the repository but require that the user be logged in before they be activated. In general, authorizations become available when the associated person logs in and satisfies a proof of identity protocol, presuming that they have not expired.
Source specifications list certificates that must be on the same repository as the digital work. These certificates become accessible whenever anyone logs onto the repository. Destination specifications list certificates on the other repository in a transaction, such as the repository intended to receive a digital work.

When a digital license is used, it is “played” by the generic digital license server which is part of the software of a repository. The digital license server is a certified program, whose validity can be assured by various means. A license itself may specify that a special purpose license server may be needed, either on the repository or accessed remotely.

Example:

```
<PLAY>
     <FEE><MONETARY>
           <PERUSE value=”15.00”/>
           <ACCOUNT><ACCOUNTTO id=”Jones-Pub248afdoiu4398”/></ACCOUNT>
     </MONETARY></FEE>
</PLAY>
<PLAY>
    <ACCESS><PRINCIPAL><CERTIFICATE>
               <AUTHORITY id="Jones Publishers"></AUTHORITY>
               <PROPERTYPAIR name="Type" value="Movie Distributor"/>
    </CERTIFICATE></PRINCIPAL></ACCESS>
</PLAY>
<PLAY>
    <ACCESS><SOURCE><CERTIFICATE>
               <AUTHORITY id="Jones Publishers"></AUTHORITY>
                     <PROPERTYPAIR name="Type" value="Site License"/>
    </CERTIFICATE></SOURCE></ACCESS>
</PLAY>
```

In the example, the digital work has three Play rights. To exercise the first Play right requires no particular authorization, but has a per use fee of fifteen dollars. To exercise the second Play right requires no fee, but requires the user to have a license issued by Jones Publishers as a movie distributor. To exercise the third right requires no fee but requires a site license on the repository.

The next three examples show source and destination specifications. Destination specifications are relevant in transaction involving two repositories, such as a Transfer transaction that moves a digital work from a source repository to a destination repository. A destination specification describes certificates that must be displayed by the destination repository. A source specification describes certificates that must be displayed by the source repository.

Example:

```
<PRINT>
     <ACCESS><DESTINATION><CERTIFICATE>
           <AUTHORITY id="Digital Property Trust"></AUTHORITY>
           <PROPERTYPAIR name="Protocol" value="Workgroup-33"/>
           <PROPERTYPAIR name="Resolution" value="Low"/>
     </CERTIFICATE></DESTINATION></ACCESS>
</PRINT>
<COPY>
     <ACCESS>
           <SOURCE><CERTIFICATE>
                 <AUTHORITY id="Jones Publishers"></AUTHORITY>
                 <PROPERTYPAIR name="Type" value="Distributor"/>
           </CERTIFICATE></SOURCE>
           <DESTINATION><CERTIFICATE>
                 <AUTHORITY id="Jones Publishers"></AUTHORITY>
                 <PROPERTYPAIR name="Type" value="Customer-Class-1"/>
           </CERTIFICATE></DESTINATION>
     </ACCESS>
</COPY>
```

An authorization specification can include more than one certificate. The Copy right in this example can be exercised by a user that has either a distributor license or a customer-class-1 license issued by Jones publishers.

Example:

```
<COPY>
     <ACCESS><PRINCIPAL type="User">
           <CERTIFICATE>
                 <AUTHORITY id="Jones Publishers"></AUTHORITY>
                 <PROPERTYPAIR name="Type" value="Distributor"/>
           </CERTIFICATE>
           <CERTIFICATE>
                 <AUTHORITY id="Jones Publishers"></AUTHORITY>
                 <PROPERTYPAIR name="Type" value="Customer-Class-1"/>
           </CERTIFICATE>
     </PRINCIPAL></ACCESS>
</COPY>
```

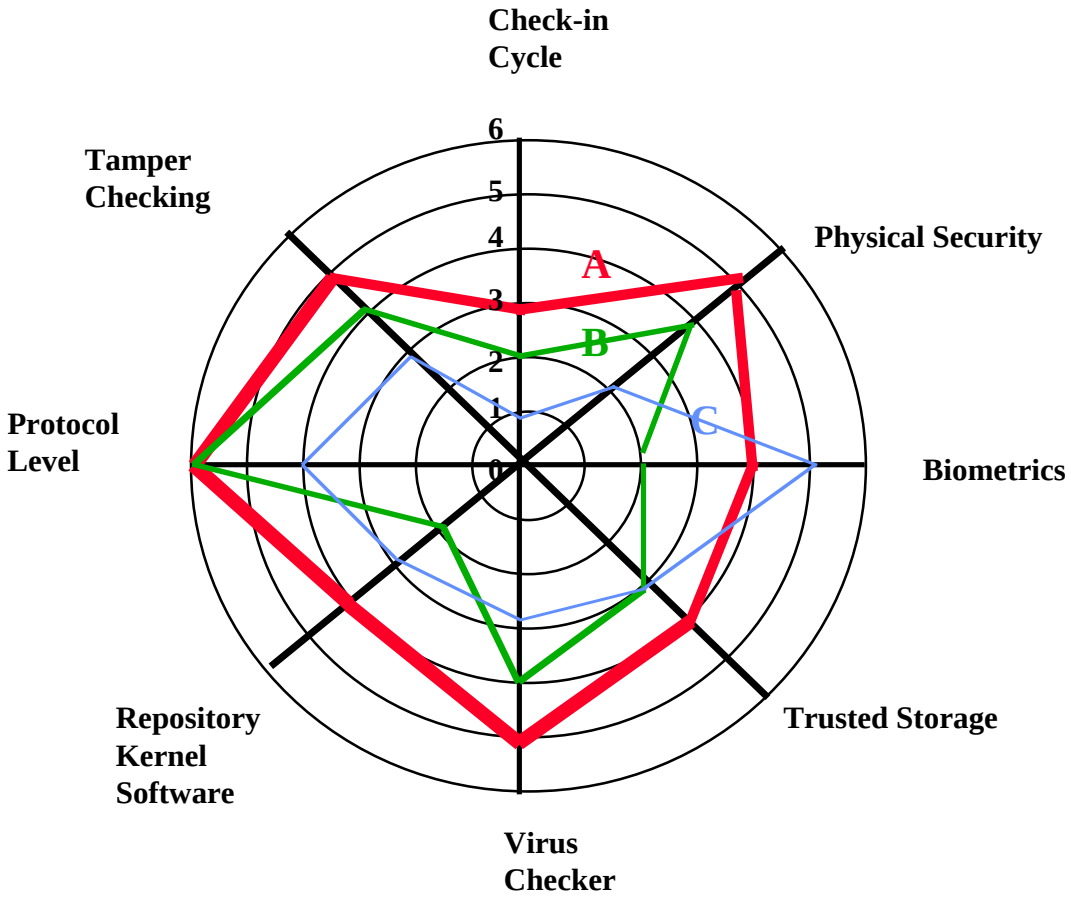
4.5.2 Security Classes

By using certificates to indicate specific system properties, it is possible to be arbitrarily precise in characterizing requirements on a transaction for a trusted system. Any particular fact to be tested can be represented in terms of a valid digital certificate that must be presented by a system.
A disadvantage of specifications that can only test for the presence or absence of particular digital certificates is that there is no easy way to specify that a publisher wants a system that is "at least this secure". From a practical perspective, there may be many ways to make a system secure enough for a publisher, and the number of ways to achieve security changes over time. New kinds of systems suitable for holding a work can appear after a digital work is published. This creates a need for a way to specify security such that given two specifications, A and B, it is possible to ask and answer the question "Does specification A describe a requirement that is at least as secure as specification B?" Facilitating such statements and comparisons is the purpose of the security classes part of the extensible rights markup language.

The design goals for security class specifications in the rights markup language are:

- It should be possible to specify different security class requirements for different rights.
- Given two security class specifications, it should be possible to determine their ranking in a partial ordering. Restated, it should be possible to determine whether specification A is at least as secure as specification B.
- The criteria for describing security provisions should admit the use of multiple dimensions, such as phyla security, communications security, defenses against viruses, method for identity testing, and so on.
- It should be possible to add new dimensions for security over time, that is, the language of dimensions should be extensible.
- Claims that a system has particular features used in determining security should be backed by digital certificates.

The basic idea of security class specifications is a kiwiet diagram. In this diagram, there is a central point and multiple axes, each of which is a numeric scale representing some dimension. A security specification is essentially a vector, giving a numeric value for a subset of the axes. Preferably, it gives a numeric value for all of the axes, but in the case that some axis is not mentioned, a value of zero for that axis is assumed. The higher the number on an axis, the greater the security claimed. The accompanying diagram gives an example with four different specifications overlayed on the figure, each portrayed in a different color. One specification A, is strictly greater than another B, if the position specified on each axis for A is greater than the specification on the corresponding axis for B. Graphically, one specification exceeds or meets another if it entirely encloses it in the diagram, that is, specification A exceeds specification B if the polygon for A creates an outer boundary that completely contains the polygon for B.



In the accompanying figure, the three polygons A, B, and C correspond to three specifications. Specification A is greater than specification B because it exceeds it on all axes. Specification A is not greater than specification C, because C is greater than A on the biometrics axis. Similarly, C is not greater than A, because A is greater than C on the physical security axis.

DTD:

```
<!ELEMENT SECURITYLEVEL EMPTY>
<!ATTLIST SECURITYLEVEL
  name          CDATA    #IMPLIED
  value         CDATA    #REQUIRED >
```

Here is an example of a security class specification. It gives the required levels for four security dimensions. Values required for other dimensions default to zero.

Example:

```
  <PLAY><ACCESS>
            <SECURITYLEVEL name="TrustedStorage" value="4"/>
            <SECURITYLEVEL name="VirusChecker" value="5"/>
            <SECURITYLEVEL name="KernelSoftware" value="5"/>
            <SECURITYLEVEL name="ProtocolLevel" value="6"/>
  </ACCESS></PLAY>
```

4.6 Specifying Territory Information

4.6.1 The TERRITORY element

The element TERRITORY is used to specify a physical location and digital domain at/in which a right can be exercised. It consists of two optional elements, LOCATION and DOMAIN. The element LOCATION specifies a physical location or address, while the element DOMAIN specifies a virtual location in the digital domain. When more than one LOCATION and DOMAIN elements are present, the right can be exercised at all of those locations.

```
DTD:

<!ELEMENT TERRITORY ( (LOCATION | DOMAIN)* )>
<!ELEMENT LOCATION (#PCDATA)>
<!ATTLIST LOCATION
  country       CDATA    #IMPLIED
  state         CDATA    #IMPLIED
  city          CDATA    #IMPLIED
  zipcode       CDATA    #IMPLIED
  street        CDATA    #IMPLIED>
<!ELEMENT DOMAIN (#PCDATA)>
<!ATTLIST DOMAIN
  url       CDATA    #IMPLIED
  id        CDATA    #IMPLIED>

Example:

 <TERRITORY>
   <LOCATION country="us" state="CA" city="El Segundo" zipcode =
"90245"/>
   <LOCATION country="jp"/>
   <DOMAIN url="13.240.109.*">Xerox El Segundo Site</DOMAIN>
 </TERRITORY>
```

4.7 Specifying Tracking Information

4.7.1 The TRACK element

The element TRACK is used to specify a list of tracking principals and necessary information that helps to track usage of a right. It consists of two optional elements, PRINCIPAL and PARAMETERS. The element PRINCIPAL specifies a principal that is responsible for the right tracking, while the element PARAMETER specifies information used in the tracking. When more than one PRINCIPAL elements are present, the right tracking should be done by all the principals.

```
DTD:

<!ELEMENT TRACK (PRINCIPAL*, PARAMETER*)>

Example:

<TRACK>
    <PRINCIPAL>
        <OBJECT type="Tracking Server">
            <ID type="Tracker ID">US1023</ID>
            <NAME>e-tracker</NAME>
            <ADDRESS type="url">
                  "http://sometrackingservice/trackme.asp"&gt;
            </ADDRESS>
         </OBJECT>
         <VERIFICATIONDATA type="url">
```

```
                <PARAMETER name="hashed-url">
                    <VALUE encoding="base64" size="90">xxzzy</VALUE>
                </PARAMETER>
                <DIGEST>
                    <ALGORITHM>SHA1</ALGORITHM>
                    <PARAMETER name="codingtype">
                       <VALUE encoding="string">surface-coding</VALUE>
                    </PARAMETER>
                       <VALUE encoding="base64"size="160">
                              OtSrhD5GrzxMeFEm8q4pQlCKWHI=</VALUE>
                </DIGEST>
            </VERIFICATIONDATA>
            <PUBLICKEY>
                <ALGORITHM>RSA-512</ALGORITHM>
                <PARAMETER name="public exponent">
                    <VALUE encoding="integer32">3</VALUE>
                </PARAMETER>
                <PARAMETER name="MODULUS">
                    <VALUE encoding="base64" size="90">
                      33845URT203987==</VALUE>
                </PARAMETER>
            </PUBLICKEY>
      </PRINCIPAL>
      <PARAMETER name="tracking support address">
             <VALUE encoding ="url">
                  http://sometrackingservice/supportme.asp
             </VALUE>
      </PARAMETER>
</TRACK>
```

4.8 Specifying Watermark Information

The term watermark historically refers to a translucent design impressed on paper during manufacture which is visible when the paper is held to the light. Because watermarks are impressed using combinations of water, heat, and pressure, they are not easy to add or alter outside of the paper factory. Watermarks are used in making letterheads and are intended to show that a document is official.
The term watermark is now used to cover a wide range of technologies for marking rendered works, including text, digital pictures, and digital audio. Some watermarks are noticeable to people and some are hidden. In some kinds of watermarks, the embedded information is human readable, but in other kinds the information can only be read by computers.

The term fingerprint is sometimes used in contrast with watermarks, to refer to marks that carry information about the end user or rendering event rather than the document or publisher. These marks are called "fingerprints" because they can be used to trace the source of a copy back to a person or computer that rendered the original. In contrast, the term watermark is often used to refer to marks that carry information to identify the work or publisher.

The same technologies and kinds of marks, e.g. glyph boxes, can be used to carry both kinds of information. In practice, it is not only possible but often desirable and convenient to combine both kinds of information — for watermarks and fingerprints — in a single mark. For simplicity in this document, we use the term watermark generally to refer to any marking technique for embedding information, whether it is about the digital work itself or the rendering event.

Different kinds of watermarks have different capacities for embedding data. For example, glyph boxes can be added to a printout of a document. They are capable of robustly encoding about 300 bytes per square inch on 300 dpi laser printers. With data compression, more information can be encoded in the same space.
Different kinds of watermarks are suitable for different kinds of works and media. For example, different techniques are used for watermarking music, movies, books, and high resolution photographs. For example, watermarks for music can embed data in the digital signal that are inaudible to the human ear when the

music is played. The digital representations for different kinds of work -- e.g. for music, video, and text -- are varied and constantly evolving.

The following are design goals for watermark capabilities in XrML:

- Publishers can specify that watermarks embed information known when a work is published (e.g. title and author) as well as information known only when a work is rendered (e.g. user name or printer name).
- The specification of what information is to be embedded in a watermark is specified in XrML statements.
- Media-dependent specifications of a watermark, such as the shape and location of a glyph box or the parameters of an acoustic watermark, are not expressed in XrML. These are expressed within the digital contents of a work or are implied by the type of rendering device. (This separation of responsibility for representation is intended to allow digital content representations and XrML to evolve independently.)
- The specification of what information is to be embedded in a watermark is separate from the specification of the type of watermark.
- The type of watermark used is implied by the type of the printer or player.

Using XrML, a publisher requests what information is to be expressed in a watermark. Since watermarks may have a limited capacity to embed information, it is possible for a publisher to request more information to be included in a watermark than can fit. The standard policy for rendering devices is to truncate the information expressed in the watermark when the capacity limit is reached. This means that the publisher can choose to express the most important information first. To the extent possible, one function of a Rights Editor is to inform a publisher of cases where information will be truncated.

In the case where more than one watermark appears in a work (or on a page in the case of trusted printing), then the information expressed in each watermark begins at the beginning. In a multimedia work the same information may be encoded by completely different techniques in pictures, music, and linked text as the work is played or printed. In this example, XrML expresses the information to be encoded and the players and printers embed it in different digital representations and renderings.

4.8.1 Watermark Strings, Tokens, and Objects

The following table gives the syntax for watermark specification in XrML.

```
DTD:

<!ELEMENT WATERMARK((WATERMARK-STR | WATERMARK-TOKENS |
                     WATERMARK-OBJECT)* )>
<!ELEMENT WATERMARK-STR EMPTY>
<!ATTLIST WATERMARK-STR
  string        CDATA    #REQUIRED>
<!ELEMENT WATERMARK-TOKENS EMPTY>
<!ATTLIST WATERMARK-TOKENS
  all-rights           (true|false) "false"
  render-rights        (true|false) "false"
  user-name            (true|false) "false"
  user-id              (true|false) "false"
  user-location        (true|false) "false"
  institution-name     (true|false) "false"
  institution-id       (true|false) "false"
  institution-location (true|false) "false"
  render-name          (true|false) "false"
  render-id            (true|false) "false"
  render-location      (true|false) "false"
  render-time          (true|false) "false"
  copy-number          (true|false) "false">
<!ELEMENT WATERMARK-OBJECT (OBJECT)>
```

Thus, a watermark specification is a list of sources of information. The elements of the list can be strings of text known at the time the work is published (W*atermark-Str*), they can be lists of tokens signifying "fingerprint" information known at the time a work is rendered *(Watermark-Token*), or they can be digital objects whose bits are to be encoded (W*atermark-Object*). The objects are assumed to present in the contents of the digital work being rendered, specified by W*orkId*s in the XrML specification.

The watermark tokens when specified will take a value "true". The meanings for the various watermark tokens ("fingerprint information") are given in the following:

`all-rights`	Listing of all rights associated with the work, expressed in XrML.
`render-rights`	Listing of all render rights associated with the work, expressed in XrML.
`user-name`	The user's name.
`user-id`	The user's id, associated with his identity certificate.
`user-location`	The user's location, associated with his identity certificate.
`institution-name`	The institution's name that owns the rendering service or rendering device.
`institution-id`	The institution's id, associated with its identity certificate.
`institution-location`	The institution's location, associated with its identity certificate.
`render-name`	The name of the rendering device (e.g. the printer name) that rendered the copy.
`render-id`	The rendering device's id, associated with its identity certificate.
`render-location`	The rendering device's location, associated with its identity certificate.
`render-time`	The time and date that the work was rendered.
`copy-number`	The number of copies of the work.

4.8.2 Examples of Watermark Specifications

In the first example, there are two rendering rights on the work, a Play right and a Print right. Only the Print right specifies watermarking. The print right specifies a particular kind of trusted printer, so the type of watermarking is determined by the type of trusted printer.

The information to be saved in the watermark includes an initial string in which the title, author, and a copyright notice are given. The next part of the information for the watermark includes the user's id, the location of his institution, the name and location of the printer, and the time that the work was printed.

Example:

```
<?XML version="1.0" encoding="UTF-8"?>
<!DOCTYPE Work SYSTEM "xrml.dtd">
<XrML>
  <WORK>
    …
   <RIGHTSGROUP name="Regular">
        <BUNDLE>
            <FEE>
                  <ACCOUNT>
                    <ACCOUNTTO id="123456789"/><HOUSE id="Visa"/>
                  </ACCOUNT>
            </FEE>
            <ACCESS>
                  <SECURITYLEVEL name="Physical" value="4"/>
                  <SECURITYLEVEL name="Kernel" value="5"/>
            </ACCESS>
        </BUNDLE>
        <RIGHTSLIST>
            <COPY><FEE><MONETARY><PERUSE value="5"/>
```

```
              </MONETARY></FEE></COPY>
            <TREANSFER></TREANSFER>
            <DELETE></DELETE>
            <PLAY></PLAY>
            <PRINT>
              <PRINTER><CERTIFICATE>
                <AUTHORITY id="DTP"></AUTHORITY>
                <PROPERTYPAIR name="Type" value="TrustedPrinter-6"/>
              </CERTIFICATE></PRINTER>
              <WATERMARK>
                <WATERMARK-STR
                  string="Title: 'Zeke Zack - The Moby Dog Story'
                    Copyright 1994 Zeke Jones All Rights Reserved"/>
                <WATERMARK-TOKENS user-id="true"
                  institution-location="true" render-name="true"
                  render-location="true" render-time="true"/>
              </WATERMARK>
              <FEE><MONETARY><PERUSE value="10"/></MONETARY></FEE>
            </PRINT>
         </RIGHTSLIST>
      </RIGHTSGROUP>
   </WORK>
</XrML>
```

The following is a second example of a watermark specification. For simplicity, it shows only the Print right. In this example, there are two strings and one set of tokens containing watermark information.

Example:

```
  <PRINT>
       <PRINTER><CERTIFICATE>
             <AUTHORITY id="DTP"></AUTHORITY>
             <PROPERTYPAIR name="Type" value="TrustedPrinter-6"/>
       </CERTIFICATE></PRINTER>
       <WATERMARK>
           <WATERMARK-STR string="Title: 'Zeke Zack - The Moby Dog Story'
                    Copyright 1994 Zeke Jones All Rights Reserved"/>
          <WATERMARK-TOKENS user-id="true" institution-location="true"
                    render-name="true" render-location="true"
                    render-time="true"/>
          <WATERMARK-STR string="Support Independent Authors!
                    Do not Photocopy this document!"/>
        </WATERMARK>
       <FEE><MONETARY><PERUSE value="10"/></MONETARY></FEE>
  </PRINT>
```

The next example is for a Play right. In this case, an acoustic watermark is used. This example differs from the previous one in that the watermark tokens include the XrML text for the render rights. Furthermore, a watermark-object is included which means that the bits of that object representation are sent to the watermarking system. One possible effect of this on an acoustic watermark, depending on the technology, is to make the "Moby Dog Jingle" play at a very low volume at some point in the music.

Example:

```
  <PLAY>
       <PLAYER><CERTIFICATE>
             <AUTHORITY id="ASCAP"></AUTHORITY>
             <PROPERTYPAIR name="Type" value="TrustedPrinter-14"/>
       </CERTIFICATE></PLAYER>
       <WATERMARK>
```

```
            <WATERMARK-STR string="Title: Moby Dog Howling Song
                  Copyright 1994 Zeke Jones All Rights Reserved"/>
            <WATERMARK-TOKENS user-id="true"
                  institution-location="true" render-rights="true"/>
            <WATERMARK-OBJECT>
                  <OBJECT><ID>Moby-Dog-Jingle-09834jvb9874</ID></OBJECT>
            </WATERMARK-OBJECT>
      </WATERMARK>
      <FEE><MONETARY><PERUSE value="10"/></MONETARY></FEE>
</PLAY>
```

4.9 Bundle Specifications

In XrML, each right can fully and separately specify the time limits, fees, access conditions, and watermark information particular for that right. This provides flexibility, but it also can lead to redundancy in the specification for those cases where many rights share the same sub-specifications. To simplify the expression of rights for those cases where rights share common terms and conditions, XrML provides bundle specifications. There can be at most one bundle specification in a rights group. For clarity, it is preferred that bundle specifications precede individual right specifications in a rights group. Programs that generate XrML should follow this convention. The syntax for a bundle specification is as follows:

```
<!ELEMENT BUNDLE (COMMENT?, %termsConditions;, WATERMARK?)
```

4.9.1 Specifying Time Limits Inside Bundles

XrML makes it possible to give separate and distinct time specifications for every right associated with a work. However, in many cases, the time specifications are the same for all of the rights in a rights group. To avoid repeating the specification for every right, a time specification can be expressed once within a *bundle* specification. In this case, the meaning of the time specification is the same as it were repeated for every right in the right group.

The following example is perhaps typical of the way that time limits can be expressed within bundle specifications. It enables the bearer to exercise any of the rights for two hours any time up to May 1, 1998, assuming that the trusted system has a high enough security class.

```
Example:

<RIGHTSGROUP name="Demo Rights">
      <COMMENT>
            Trial rights. Anyone can use the work for two hours
      </COMMENT>
      <BUNDLE>
            <TIME>
                  <INTERVAL hours="2"/><UNTIL>1998-05-01T8</UNTIL>
            </TIME>
            <ACCESS>
                  <SECURITYLEVEL name="Physical" value="4"/>
                  <SECURITYLEVEL name="Kernel" value="5"/>
            </ACCESS>
      </BUNDLE>
      <RIGHTSLIST>
            <LOAN></LOAN>
            <TREANSFER></TREANSFER>
            <PLAY></PLAY>
      </RIGHTSLIST>
</RIGHTSGROUP>
```

When a time specification is given in a *bundle* and a further time specification is given for some right in the rights group, the specification given for the individual right dominates. In that case, the time-specification given in the bundle is specialized by the time-specification in the right.

The next example shows how a time specification can be given in a bundle, and specialized in a right. In this case, an expiration period is given in the bundle causing all rights to expire in 1998. Furthermore, the Play right specializes this specification by indicating that the work can be played for up to two hours in this period.

```
Example:

<RIGHTSGROUP name="Demo Rights">
        <COMMENT>" Expires 1998."</COMMENT>
        <BUNDLE>
            <TIME>
                 <INTERVAL hours=”2”/><UNTIL>1998-05-01T8</UNTIL>
            </TIME>
            <ACCESS>
                <SECURITYLEVEL name="Physical" value="4"/>
                <SECURITYLEVEL name="Kernal" value="5"/>
            </ACCESS>
        </BUNDLE>
        <RIGHTSLIST>
            <LOAN></LOAN>
            <TREANSFER> </TREANSFER>
            <PLAY>
            <TIME><Metered days=”2”/><UNTIL>2050-01-01</UNTIL></TIME>
      </PLAY>
        </RIGHTSLIST>
</RIGHTSGROUP>
```

4.9.2 Specifying Fees Inside Bundles

XrML makes it possible to specify a separate fee for each individual right associated with a digital work. Sometimes it is convenient to express fee information for an entire set of rights *bundle* specification. In the most typical usage, fee specifications inside bundles are used to indicate parts of a fee specification that are common to all of the rights in the rights group. In the following example, the account number for all payments is given in the bundle-spec.

```
Example:

<RIGHTSGROUP name="Print & Play">
        <COMMENT>"All Royalties go to Jones"</COMMENT>
        <BUNDLE><FEE><MONETARY>
            <ACCOUNT>
                 <ACCOUNTTO id="123456789"/><HOUSE id="Visa"/>
            </ACCOUNT>
        </MONETARY></FEE></BUNDLE>
        <RIGHTSLIST>
            <PRINT><FEE><MONETARY>
                 <PERUSE value="5"/>
            </MONETARY></FEE></PRINT>
            <PLAY><FEE><MONETARY>
                 <PERUSE value="1"/>
            </MONETARY></FEE></PLAY>
        </RIGHTSLIST>
    </RIGHTSGROUP>
```

In the next example, a metered ticket is used to give a two hour maximum usage across all kinds of players.

```
Example:

<RIGHTSGROUP name="Demo 1999">
    <BUNDLE>
```

```
      <TIME><UNTIL>2000-01-01</UNTIL></TIME>
       <FEE><TICKET type="Two-Hour-Prepaid >
                 <AUTHORITY id="Jones Publishers"></AUTHORITY>
       </TICKET></BundleFEE>
    </BUNDLE>
    <RIGHTSLIST>
       <PLAY><PLAYER><CERTIFICATE>
          <AUTHORITY id="Consumer Products Associationn"></AUTHORITY>
          <PROPERTYPAIR name="Type" value="Big Screen 91"/>
       </CERTIFICATE></PLAYER></PLAY>
       <PLAY><PLAYER><CERTIFICATE>
          <AUTHORITY id="Consumer Products Associationn"></AUTHORITY>
           <PROPERTYPAIR name="Type" value="Small Screen 44"/>
        </CERTIFICATE></PLAYER></PLAY>
    </RIGHTSLIST>
</RIGHTSGROUP>
```

Tickets can be used across different kinds of rights. Suppose for example that a publisher has an every day price with separate fees to print or to view a document. In every day prices, it costs four dollars to print a document and ten cents an hour to view it. However, they want to offer a special rate for prepaid customers where the four dollars gives the right to print it once and also the right to view it for several hours for free.

This combination is accomplished by having two rights groups. The first rights group (“Every Day Prices”) establishes the pay as you go regular rates, and the second rights group (“PrintNView Deal”) ties printing and playing to the use of a prepaid digital ticket delivered with the digital work which costs four dollars and which authorizes printing once and unlimited viewing.

Example:

```
<RIGHTSGROUP name="Every Day Prices">
    <BUNDLE>
       <TIME><UNTIL>2000-01-01</UNTIL></TIME>
      <ACCESS>
          <SECURITYLEVEL name="Physical" value="4"/>
          <SECURITYLEVEL name="Kernal" value="5"/>
      </ACCESS>
      <FEE><MONETARY><ACCOUNT>
            <ACCOUNTTO id="Account-Jones-23475"/>
      </ACCOUNT></MONETARY></FEE>
    </BUNDLE>
    <RIGHTSLIST>
       <PLAY>
          <FEE><MONETARY><METEREDFEE>
                <RATE value=".10"></RATE><PER minutes="1"/>
           </METEREDFEE></MONETARY></FEE>
       </PLAY>
       <PRINT>
          <PRINTER><CERTIFICATE>
               <AUTHORITY id="DPT"></AUTHORITY>
               <PROPERTYPAIR name="Type" value="TrustedPrinter-5"/>
          </CERTIFICATE></PRINTER>
          <FEE><MONETARY><PERUSE value="4"/></MONETARY></FEE>
       </PRINT>
    </RIGHTSLIST>
  </RIGHTSGROUP>
  <RIGHTSGROUP name="PrintNView Deal">
      <BUNDLE>
         <TIME><UNTIL>2000-01-01</UNTIL></TIME>
         <ACCESS>
                <SECURITYLEVEL name="Physical" value="4"/>
```

```
                <SECURITYLEVEL name="Kernal" value="5"/>
            </ACCESS>
            <FEE><TICKET type="PrintNView">
                    <AUTHORITY id="Jones Publishers"></AUTHORITY>
            </TICKET></FEE>
        </BUNDLE>
        <RIGHTSLIST>
            <PLAY></PLAY>
            <PRINT><PRINTER><CERTIFICATE>
                    <AUTHORITY id="DPT"></AUTHORITY>
                    <PROPERTYPAIR name="Type" value="TrustedPrinter-5"/>
                </CERTIFICATE></PRINTER></PRINT>
        </RIGHTSLIST>
    </RIGHTSGROUP>
```

4.9.3 Specifying Access Inside Bundles

XrML makes it possible to give separate and distinct access specifications for every right associated with a work. In many cases, the access specifications are the same for all rights in a rights group, or they may have many elements in common. To simplify specifications for these common cases, an access specification can be included within a *bundle* specification. In this case, the meaning is the same as if the same time specification were repeated for every right in the right group.

When an access specification is given in a *bundle* specification and also another access specification is given for some right in the rights group, the specification given for the individual right dominates.
In that case, the time-specification given in the bundle is ignored for that right in the interpretation process. In the following example, all of the rights require security class 3 or greater.

Example:

```
 <RIGHTSGROUP name="Standard Set">
        <BUNDLE><ACCESS>
            <SECURITYLEVEL name="Physical" value="4"/>
           <SECURITYLEVEL name="Kernal" value="5"/>
        </ACCESS></BUNDLE>
        <RIGHTSLIST>
             <LOAN></LOAN>
             <TREANSFER></TREANSFER>
             <PLAY></PLAY>
        </RIGHTSLIST>
 </RIGHTSGROUP>
```

4.9.4 Specifying Watermark Information Inside Bundles

XrML makes it possible to give separate and distinct specifications for watermark information for every Print or Play right in a digital work. In many cases, the watermark information is the same for all Print and Play rights in a group. To simplify specifications for these common cases, a watermark specification can be included within a *bundle* specification. In this case, the meaning is the same as if the same watermark specification were repeated for every Print and Play right in the right group.

Example:

```
<?XML version="1.0" encoding="UTF-8"?>
<!DOCTYPE Work SYSTEM "xrml.dtd">
<XrML>
  <WORK>
    …
    <RIGHTSGROUP name="Regular">
        <BUNDLE>
            <ACCESS>
                <SECURITYLEVEL name="Physical" value="4"/>
```

```
                <SECURITYLEVEL name="Kernal" value="5"/>
            </ACCESS>
            <FEE><MONETARY><ACCOUNT>
                <ACCOUNTTO id="123456789"/><HOUSE id="MasterCard"/>
            </ACCOUNT></MONETARY></FEE>
            <WATERMARK>
                <WATERMARK-STR string="Title: 'Moby Dog'
                  Copyright 1994 by Zeck Jones. All Rights Reserved."/>
                <WATERMARK-TOKENS user-id="true"
                  institution-location="true" render-name="true"
                  render-location="true" render-time="true"/>
             </WATERMARK>
        </BUNDLE>
        <RIGHTSLIST>
            <COPY>
                <FEE><MONETARY><PERUSE value="5"/></MONETARY></FEE>
            </COPY>
            <TREANSFER></TREANSFER>
            <DELETE></DELETE>
            <PLAY></PLAY>
            <PRINT>
                <PRINTER><CERTIFICATE>
                    <AUTHORITY id="DPT"></AUTHORITY>
                    <PROPERTYPAIR name="Type" value="TrustedPrinter-6"/>
                </CERTIFICATE></PRINTER>
                <FEE><TICKET type="Prepaid-Pri">
                    <AUTHORITY id="Jones Publishers"></AUTHORITY>
                </TICKET></FEE>
             </PRINT>
             <PRINT>
                <PRINTER><CERTIFICATE>
                    <AUTHORITY id="DPT"></AUTHORITY>
                    <PROPERTYPAIR name="Type" value="TrustedPrinter-6"/>
                </CERTIFICATE></PRINTER>
                <FEE><MONETARY><PERUSE value = "10"/></MONETARY></FEE>
             </PRINT>
          </RIGHTSLIST>
      </RIGHTSGROUP>
  </WORK>
</XrML>
```

In this example, the rights group contains a bundle with fee, access, and watermark specifications. Watermark specifications apply to all render right statements in the rights group. In the example, they apply to the Play right and to both Print rights.

When a watermark specification is given for a printer or player that does not support watermarks, it is ignored. In this example, if the player does not embed watermarks, the specification will be effective only for the two rights involving trusted printers.

5 RULES FOR GENERATING AND PROCESSING XRML DOCUMENTS

Broadly construed, XrML is about "terms and conditions" of use and their integrity. Terms and conditions ultimately refer to facts about the world and permitted actions on digital works. Examples of facts are: the current date and time, whether the user is a member of the book of the month club, and the security class of the trusted system. XrML requires well-defined and certified ways of determining such facts, such as trusted clocks and accessible, signed digital certificates attesting to various facts. Restated, the kinds of facts XrML uses to determine whether a right can be exercised are always determinable in a reliable way by accessing digital certificates and controlled internal parameters of trusted systems.

Actions sanctioned by terms and conditions are always well-defined transactions corresponding to exercising specific digital rights. Thus, there are transactions defined for such actions as printing, copying, or making an encrypted backup copy. All of these actions are performed by trusted systems. There are no rights in XrML for actions performed by people, such as "play for educational purposes," although this might alternatively be approximated as "play for a user possessing a specific, valid educator or student certificate."

<<The rest of this section is under construction. Editor>>

6 DIFFERENCES WITH DPRL

This section lists differences between XrML and DPRL 2.0. Basically, XrML is an extension of DPRL; it now includes information about who issues a XrML document and how the integrity of the document can be verified via digital signatures.

❖ Changed Top-Level Document Structure

The top-level element of a DPRL document is `WORK`. A XrML document now has the following top-level structure:

```
<XrML>
  <BODY>
        (ISSUED)?
        (TIME)?
        (DESCRIPTOR)?
        (ISSUER)?
        (ISSUEDPRINCIPALS)?
        (WORK)?
  </BODY>
  (SIGNATURE)?
</XrML>
```

❖ Changed elements FROM and UNTIL

The definition of element `FROM` is changed from

```
<!ELEMENT FROM EMPTY>
<!ATTLIST FROM %moment>
```

to

```
<!ELEMENT FROM (%moment;)>
```

A similar change is also applied to element `UNTIL`. This is due to the change of definition of time moment from a list of attributes to a single string in the format of "yyyy-mm-ddThh:mm:ss".

❖ Added VIEW right

For convenience, a new render right `VIEW` is added.

❖ Removed elements BUNDLEFEE and BUNDLEFEE

The elements `BUNDLEFEE` and `BUNDLEMONETORY` in DPRL are removed, because they can be replaced by the elements `FEE` and `MONETORY`.

❖ Changed element TICKET

The definition of `TICKET` is changed from

```
<ELEMENT TICKET (AUTHORITY, TYPE)>
```

to

```
<ELEMENT TICKET (AUTHORITY)>
<!ATTLIST TICKET
  type     CDATA    #REQUIRED
  id       CDATA    #REQUIRED >
```

❖ Changed ANY from a named principal to a principal type

The named principal `ANY` in access control conditions is removed. The following can be used to replace it.
<PRINCIPAL type = "any"> … </PRINCIPAL>

❖ Changed element AUTHORITY

The definition of `AUTHORITY` is changed from

```
<ELEMENT AUTHORITY EMPTY>
<!ATTLIST AUTHORITY
```

```
  id     CDATA    #REQUIRED >
```
to
```
<ELEMENT AUTHORITY EMPTY>
<!ATTLIST AUTHORITY
  type     CDATA    #REQUIRED
  id       CDATA    #REQUIRED >
```

❖ Added TRACK condition

To enable tracking work usage, a new element TRACK is added to the terms and conditions. It specifies tracking principals and related parameters.

```
<!ELEMENT TRACK (PRINCIPAL*, PARAMETER*) >
```

❖ Added TERRITORY condition

To enable restriction on where a work can be used, a new element TERRITORY is added to the terms and conditions. It specifies physical locations and/or digital domains at which a right can be exercised.

```
<!ELEMENT TERRITORY ( (LOCATION | DOMAIN)* ) >
```

❖ Changed attribute work-id of element WORK to id

To be consistent with other attributes, the attribute WORK-ID of element WORK is changed to ID.

❖ Changed attribute copy-count of elements PRINT and COPY to maxcount

To avoid possible confusion, the attribute COPY-COUNT of elements PRINT and COPY is changed to MAXCOUNT.

❖ Changed attribute name of element RIGHTSGROUP to name

To simply the matter, the attribute NAME of element RIGHTSGROUP is changed to NAME.

❖ Changed element NEXTCOPYRIGHTS in transport rights to NEXTRIGHTS

To simply the matter and avoid possible confusion, the element NEXTCOPYRIGHTS in all transport rights is changed to NEXTRIGHTS.

❖ Changed element backupcopyrights in file management rights to backuprights

To simply the matter and avoid possible confusion, the element BACKUPCOPYRIGHTS in all file management rights is changed to BACKUPRIGHTS.

❖ Changed attribute id of elements ACCOUNTFROM and ACCOUNTTO to id

To simply the matter, the attribute ID of elements Accountfrom and ACCOUNTTO is changed to ID.

❖ Changed attribute id of element HOUSE to id

To simply the matter, the attribute ID of element HOUSE is changed to ID.

❖ Added attributes type and url to elements ACCOUNTFROM, ACCOUNTTO and HOUSE

To indicate type and location of accounting firms, the attributes TYPE and URL are added to elements Accountfrom, ACCOUNTTO and HOUSE.

❖ Renamed element security to securitylevel

To be precise and accurate, the element Security is renamed to Securitylevel.

❖ Added attributes type and url to elements ACCOUNTFROM, ACCOUNTTO and HOUSE

To indicate type and location of accounting firms, the attributes TYPE and URL are added to elements Accountfrom, ACCOUNTTO and HOUSE.

❖ Removed element RightsVersion

The element RIGHTSVERSION in DPRL is removed. The attribute version of the element BODY resumes its functionality.

❖ Removed element WorkId

As a work is identified using an OBJECT element, the element WORKID in DPRL is removed. The ID element in OBJECT is used as an identification of the work.

❖ Added elements Creator and Digest

Two new elements CREATOR and DIGEST are added as component elements of the element WORK.

❖ Changed element OWNER to be a named principal

The element OWNER is now defined as a named principal.

❖ Changed attribute group-name to name

To simply the matter, the attribute group-name of the elements RIGHTSGROUP, REFERENCERIGHTSGROUP, NETXTRIGHT, REMAININGRIGHTS, RIGHTSTOADD and RIGHTSTODELETE is changed to name.

7 APPENDIX A. LEXICAL CONVENTIONS

This section explains conventions for describing elements of the rights markup language.
Blanks, horizontal and vertical tabs, new lines, form feeds, and comments are referred to as “white space” and serve to separate tokens. White space or punctuation is required to separate adjacent identifiers, keywords and constants. The following characters are used as punctuation:

< > “ ”

No distinction is made between left and right double quotes. There are three kinds of tokens: keywords, constants and identifiers.

7.1 Keywords

The following are keywords in the rights markup language. This list does not include the names of foreign currencies or watermark tokens, which are listed separately. It also does not include property names that may become standardized for certain classes of digital certificates. The XML keywords are also not mentioned in this list.

Access
Account
AccountFrom
AccountTo
Any
Authority
Backup
BackupCopyRights
BestPriceUnder
Bundle
BundleFee
BundleMonetary
By
CallForPrice
Certificate
Comment
Contents
Copies
Copy
Currency
Delete
Description
Destination
Directory
Edit
Editor
Embed
Export
Extract
Fee
Folder
From
House
Install
Interval
Loan
Markup
Max
Metered
MeteredFee
Min
Monetary
NextCopyRights
Owner
Parts
Per
PerUse
Play
Player
Print
Printer
PropertyPair
Rate
ReferencedRIGHTSGROUP
RemainingRights
Repository
Restore
RIGHTSGROUP
RightsList
RightsToAdd
RightsToDelete
RightsVersion
Security
Source
Ticket
Time
Transfer
Type
Uninstall
Until
User
Verifier
Verify
Watermark
Watermark-Object
Watermark-Str
Watermark-Tokens
Work
WorkId

7.2 Constants

In XML literal, string, integer, floating constants are all considered to be character data(CDATA). It is up to the parser to check for the validity of the data.

Character data is any string of characters which does not contain the start delimiter ”<” enclosed within double quotes(“). A character is an atomic unit of text as specified by ISO/IEC10646. Legal characters are tab, carriage return, line feed, and the legal graphic characters of Unicode and ISO/IEC 10646.

7.3 Identifiers

Identifiers are the attributes values in elements. All characters are significant, and upper- and lower-case characters are distinct. The following are the name of identifiers in the rights markup language.

id	institution-name	sec
all-rights	interval	seconds
id		string
id	iso-currency-code	ticket-id
copy-count	min	to
	minutes	user-id
copy-number	mm	user-location
days	moment	user-name
dd	months	value
dealer-id	name	
from	percentage	work-id
name	render-id	
hours	render-location	years
hrs	render-name	yyyy
institution-id	render-rights	
institution-location	render-time	

Explained below is the meaning of some of the key identifiers of XrML.

An *id* is an identifier for a specific account at a specific bank or other financial institution from which fees will be collected, or to which incentives will be paid.

An *id* is an identifier for an organization that issues digital certificates.

An *id* is an identifier for a financial clearing house, such as a bank or credit card consortium.

A *id* is an identifier of a digital certificate that is used to grant permission for a one-time transaction on a particular work.

A *id* is an identifier of a digital work.

These identifiers are always contained in digital certificates, signed by an issuer. Conventions and procedures for establishing meaning for these id's is beyond the scope of this manual.

7.4 Currency Codes

For compatibility with other standards, XrML uses the ISO 4217 three-character currency code. In most cases, the ISO 4217 currency code is composed of the country's two-character ISO 3166 country code plus a one-character currency designator. More details are given on the ISO On-line home page.

ADP Andorran Peseta
AED United Arab Emirates Dirham
AFA Afghanistan Afghani
ALL Albanian Lek
ANG Netherlands Antillian Guilder
AOK Angolan Kwanza
ARA Argentinian Austral
ATS Austrian Schilling
AUD Australian Dollar
AWG Aruban Florin
BBD Barbados Dollar
BDT Bangladeshi Taka
BEF Belgian Franc
BGL Bulgarian Lev
BHD Bahraini Dinar
BIF Burundi Franc
BMD Bermudian Dollar
BND Brunei Dollar
BOB Bolivian Boliviano
BRC Brazilian Cruzeiro
BSD Bahamian Dollar
BTN Bhutan Ngultrum
BUK Burma Kyat
BWP Botswanian Pula
BZD Belize Dollar
CAD Canadian Dollar
CHF Swiss Franc
CLF Chilean Unidades de Fomento
CLP Chilean Peso
CNY Yuan (Chinese) Renminbi
COP Colombian Peso
CRC Costa Rican Colon
CSK Czech Koruna
CUP Cuban Peso
CVE Cape Verde Escudo
CYP Cyprus Pound
DDM East German Mark (DDR)
DEM Deutsche Mark
DJF Djibouti Franc
DKK Danish Krone
DOP Dominican Peso
Algerian Dinar
ECS Ecuador Sucre
EGP Egyptian Pound
ESP Spanish Peseta
ETB Ethiopian Birr
FIM Finnish Markka
FJD Fiji Dollar
FKP Falkland Islands Pound
FRF French Franc
GBP British Pound
GHC Ghanaian Cedi
GIP Gibraltar Pound
GMD Gambian Dalasi
GNF Guinea Franc
GRD Greek Drachma
GTQ Guatemalan Quetzal
KRW (South) Korean Won
KWD Kuwaiti Dinar
KYD Cayman Islands Dollar
LAK Lao Kip
LBP Lebanese Pound
LKR Sri Lanka Rupee
LRD Liberian Dollar
LSL Lesotho Loti
LUF Luxembourg Franc
LYD Libyan Dinar
MAD Moroccan Dirham
MGF Malagasy Franc
MNT Mongolian Tugrik
MOP Macau Pataca
MRO Mauritanian Ouguiya
MTL Maltese Lira
MUR Mauritius Rupee
MVR Maldive Rufiyaa
MWK Malawi Kwacha
MXP Mexican Peso
MYR Malaysian Ringgit
MZM Mozambique Metical
NGN Nigerian Naira
NIC Nicaraguan Cordoba
NLG Dutch Guilder
NOK Norwegian Kroner
NPR Nepalese Rupee
NZD New Zealand Dollar
OMR Omani Rial
PAB Panamanian Balboa
PEI Peruvian Inti
PGK Papua New Guinea Kina
PHP Philippine Peso
PKR Pakistan Rupee
PLZ Polish Zloty
PTE Portuguese Escudo
PYG Paraguay Guarani
QAR Qatari Rial
ROL Romanian Leu
RWF Rwanda Franc
SAR Saudi Arabian Riyal
SBD Solomon Islands Dollar
SCR Seychelles Rupee
SDP Sudanese Pound
SEK Swedish Krona
SGD Singapore Dollar
SHP St. Helena Pound
SLL Sierra Leone Leone
SOS Somali Schilling
SRG Suriname Guilder
STD Sao Tome and Principe Dobra
SUR USSR Rouble
SVC El Salvador Colon
SYP Syrian Potmd
SZL Swaziland Lilangeni
THB Thai Bhat
TND Tunisian Dinar

GWP Guinea-Bissau Peso
GYD Guyanan Dollar
HKD Hong Kong Dollar
HNL Honduran Lempira
HTG Haitian Gourde
HUF Hungarian Forint
IDR Indonesian Rupiah
IEP Irish Punt
ILS Israeli Shekel
INR Indian Rupee
IQD Iraqi Dinar
IRR Iranian Rial
ISK Iceland Krona
ITL Italian Lira
JMD Jamaican Dollar
JOD Jordanian Dinar
JPY Japanese Yen
KES Kenyan Schilling
KHR Kampuchean (Cambodian) Riel
KMF Comoros Franc
KPW North Korean Won
TOP Tongan Pa'anga
TPE East Timor Escudo
TRL Turkish Lira
TTD Trinidad and Tobago Dollar
TWD Taiwan Dollar
TZS Tanzanian Schilling
UGS Uganda Shilling
USD US Dollar
UYP Uruguayan Peso
VEB Venezualan Bolivar
VND Vietnamese Dong
VUV Vanuatu Vatu
WST Samoan Tala
YDD Democratic Yemeni Dinar
YER Yemeni Rial
YUD New Yugoslavia Dinar
ZAR South African Rand
ZMK Zambian Kwacha
ZRZ Zaire Zaire
ZWD Zimbabwe Dollar

8 APPENDIX B. GRAMMAR FOR THE DIGITAL PROPERTY LANGUAGE

This section collects together the complete DTD for the eXtensible rights Markup Language (XrML).

```
<!--
    Extensible rights Markup Language 1.03 DTD

    Name space = http://www.xrml.org/xrml

    For further information, see http://www.xrml.org

    Copyright (c) 2000 ContentGuard, Inc.
    All Rights Reserved

    $Revision: 1.0 $
    $Date: 2000/04/27 07:31:24 $

    $Revision: 1.02 $
    $Date: 2000/06/07 12:05:12 $
    - Introduce entity "moment", and use it in "ISSUED", "FROM" and
"UNTIL".
    - Add element AUTHENTICATEDDATA to element BODY
    - Removed element PRINCIPAL from element ENABLINGBITS
    - Added element PRINCIPAL to element PRINCIPAL

    $Revision: 1.03 $
    $Date: 2000/06/09 15:32:04 $
    - Add "SECURITYLEVEL" to "PRINCIPAL"
-->


<!--================= General Entities =================-->

<!-- a Uniform Resource Identifier, see [RFC2396] -->
<!ENTITY % URI  "CDATA">

<!-- Time Moment Entity -->
<!-- the time format is "yyyy-mm-ddThh:mm:ss"; e.g., 2000-01-27T15:30 --
>
<!ENTITY % moment "#PCDATA" >

<!-- Interval Entity -->
<!-- time interval in terms of
     years, months, days, hours, minutes, and seconds
-->
<!ENTITY % interval
  "years        CDATA    #IMPLIED
   months       CDATA    #IMPLIED
   days         CDATA    #IMPLIED
   hours        CDATA    #IMPLIED
   minutes      CDATA    #IMPLIED
   seconds      CDATA    #IMPLIED"
  >

<!-- Principal Entity -->
<!-- generic entity with
     object            - principal object: id, name, etc.
     authenticator     - principal that authenticates this principal
```

```
     publickey         - public key of this principal
     privatekey        - private key of this principal
     digest            - digest value needed to ensure
                         identity of this principal
     verificationdata  - verification data needed to ensure
                         identity of this principal
     enablingbits      - extra (secret) information associated
                         with this principal
     certificate       - certificate of this principal
     principal         - principal whose credential and keys are
                         used to seal and unseal the element
ENABLINGBITS.
     securitylevel     - security level of this principal
-->
<!ENTITY % aPrincipal
  "OBJECT?,
   (AUTHENTICATOR    |
    PUBLICKEY        |
    PRIVATEKEY       |
    DIGEST           |
    VERIFICATIONDATA |
    ENABLINGBITS     |
    CERTIFICATE      |
    PRINCIPAL        |
    SECURITYLEVEL )*"
  >


<!-- TermsConditions Entity -->
<!-- terms and conditions associated with a usage right
     time      - when the right can be exercised
     access    - under what conditions the right can be exercised
     fee       - for what fees if the right is exercised
     territory - at which location the right can be exercised
     track     - where and how to track the usage
-->
<!ENTITY % termsConditions
  "(TIME      |
    ACCESS    |
    FEE       |
    TERRITORY |
    TRACK)+"
  >


<!--================= Document Structure =================-->
<!-- An XrML spec consists of a body and
     possibly a digital signature of the body
-->
<!ELEMENT XrML (BODY,
                SIGNATURE?)
  >
<!ATTLIST XrML
  xrmlns    %URI;    #FIXED 'http://www.xrml.org/xrml'
  >

<!--================= Body Structure =================-->
<!-- The BODY element consists of possibly the following components:

     TIME               time interval in which this spec is valid
     ISSUED             time moment at which this spec is issued
```

```
     DESCRIPTOR          description or meta data of this spec
     ISSUER              principal who issues this spec
     ISSUEDPRINCIPALS    list of principals this spec is issued to
     WORK                work and rights this spec specifies
     AUTHENTICATEDDATA   data that provided to application

     The attribute, namespace URI, designates the body type and
     version profile. Here are two examples:

     <BODY type ="license" version="1.0" xrmlns="x-
Schema:license1Schema.xml">
     <BODY type ="work" version="2.0" xrmlns="x-Schema:work2Schema.xml">

     This makes it possible to customize XrML to a specific definition
     such as license or work.
-->

<!ELEMENT BODY (ISSUED?,
                TIME?,
            DESCRIPTOR?,
            ISSUER?,
            ISSUEDPRINCIPALS?,
            WORK?,
                AUTHENTICATEDDATA?)
  >
<!ATTLIST BODY
  type      CDATA      #IMPLIED
  version   CDATA      #IMPLIED
  xrmlns    %URI;      #IMPLIED
  >


<!--================= Time =================-->
<!-- See below in Terms and Conditions under Rights Group -->

<!-- From -->
<!-- See below in Terms and Conditions under Rights Group -->

<!-- Until -->
<!-- See below in Terms and Conditions under Rights Group -->


<!--================= Issued Time =================-->
<!ELEMENT ISSUED ( %moment;)>


<!--================= Descriptor =================-->

<!ELEMENT DESCRIPTOR (OBJECT)>


<!--================= Issuer =================-->
<!-- principal who issues this XrML spec -->
<!ELEMENT ISSUER  (%aPrincipal;)>
<!ATTLIST ISSUER
  type          CDATA    #IMPLIED
  sequence      CDATA    #IMPLIED
  >


<!--================= Issued Principals =================-->
```

```
<!-- list of principals to whom this XrML spec is issued to -->
<!ELEMENT ISSUEDPRINCIPALS (PRINCIPAL+)>


<!--================= Object =================-->
<!-- generic entity for identifying some object
     id      - identification of the object
     name    - name of the object
     address - location of the object
-->
<!ELEMENT OBJECT (ID,
                  NAME?,
                  (ADDRESS*)?)
  >
<!ATTLIST OBJECT
  type           CDATA     #IMPLIED
  version        CDATA     #IMPLIED
  >

<!-- Object Id -->
<!ELEMENT ID (#PCDATA)>
<!ATTLIST ID
  type           CDATA     #REQUIRED
  version        CDATA     #IMPLIED
  anonymitylevel CDATA     #IMPLIED
  >

<!-- Object Name -->
<!ELEMENT NAME (#PCDATA)>

<!-- Object Address -->
<!ELEMENT ADDRESS (#PCDATA)>
<!ATTLIST ADDRESS
  type           CDATA      #REQUIRED
  >


<!--================= Principal =================-->
<!-- General entity for a principal with attributes
     type        - type of the principal, e.g., "cpu", "user"
     internal-id - reference to its index in the ISSUEDPRINCIPALS list
-->
<!ELEMENT PRINCIPAL  (%aPrincipal;) >
<!ATTLIST PRINCIPAL
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED
  >


<!--================= Authenticator =================-->
<!-- A principal who authenticates other principals
     ID                  - id of the authenticator
     NAME                - name of the authenticator
     AUTHENTICATOR       - an authenticator who authenticates the
                           authenticator; This enables an authentication
                           chain.
     AUTHENTICATIONCLASS - classification of the authenticator
     VERIFICATIONDATA    - information the authenticator uses to
authenticate
                           principals; there could be multiple
VERIFICATIONDATA
```

```
     type                - type of the authenticator
     internal-id         - reference to its index in the
ISSUEDPRINCIPALS list
-->
<!ELEMENT AUTHENTICATOR (ID,
                   NAME?,
                         AUTHENTICATOR?,
                     AUTHENTICATIONCLASS?,
                     VERIFICATIONDATA+)
  >
<!ATTLIST AUTHENTICATOR
  type          CDATA     #REQUIRED
  internal-id   CDATA     #IMPLIED
  >

<!-- AuthenticationClass -->
<!ELEMENT AUTHENTICATIONCLASS ( (VERSION     |
                                 VERSIONSPAN |
                                 SECURITYLEVEL)*)
  >

<!-- Version -->
<!-- current version -->
<!ELEMENT VERSION (#PCDATA)>

<!-- Version Span -->
<!-- versions covered -->
<!ELEMENT VERSIONSPAN EMPTY>
<!ATTLIST VERSIONSPAN
  min            CDATA     #IMPLIED
  max            CDATA     #IMPLIED
  >

<!-- VerificationData -->
<!ELEMENT VERIFICATIONDATA ( (PUBLICKEY |
                              PRIVATEKEY |
                              PARAMETER |
                              DIGEST)* )
  >
<!ATTLIST VERIFICATIONDATA
  type          CDATA     #REQUIRED
  >

<!-- Public Key -->
<!ELEMENT PUBLICKEY (ALGORITHM,
                     PARAMETER*)
  >

<!-- Private Key -->
<!ELEMENT PRIVATEKEY (ALGORITHM,
                      PARAMETER*)
  >

<!-- Algorithm -->
<!ELEMENT ALGORITHM (#PCDATA)>

<!-- Parameter -->
<!ELEMENT PARAMETER (VALUE)>
<!ATTLIST PARAMETER
  name          CDATA     #REQUIRED
  >
```

```
<!-- Value -->
<!ELEMENT VALUE (#PCDATA)>
<!ATTLIST VALUE
  encoding      CDATA     #REQUIRED
  size          CDATA     #IMPLIED
  >

<!-- Enabling Bits -->
<!-- information that enables a principal to use this XrML doc or
     any other information (e.g., work) it is associated with.
     VALUE     - value of the enabling bits
     PRINCIPAL - a principal whose private key can unseal the enabling
bits
     type      - type of the enabling bits
-->
<!ELEMENT ENABLINGBITS (VALUE)>
<!ATTLIST ENABLINGBITS
  type          CDATA     #REQUIRED
  >


<!--================= Signature =================-->
<!-- Digital signature of the BODY of an XrML doc
     DIGEST    - Digest of the BODY
     ALGORITHM - Algorithm for digital signature
     PARAMETER - parameters for the algorithm
     VALUE     - value of the digital signature
-->
<!ELEMENT SIGNATURE (DIGEST,
                     ALGORITHM?,
                     PARAMETER*,
                     VALUE)
  >


<!--================= Digest =================-->
<!-- Digital digest of some message
     ALGORITHM  - algorithm for generating digest
     PARAMETER  - parameters for the digest algorithm
     VALUE      - digest value
     sourcedata - source data the digest value is generated from
-->
<!ELEMENT DIGEST (ALGORITHM,
                  PARAMETER*,
                  VALUE?)
  >
<!ATTLIST DIGEST
  sourcedata    CDATA     #IMPLIED
  >

<!-- AUTHENTICATEDDATA -->
<!ELEMENT AUTHENTICATEDDATA (#PCDATA)>
<!ATTLIST AUTHENTICATEDDATA
    size CDATA  #REQUIRED
    name CDATA  #REQUIRED
  >


<!--================= Work =================-->
<!-- A work this XrML is associated with
```

```
     OBJECT                - object used to identify the work
     DESCRIPTION           - description of the work
     CREATOR               - creator of the work
     OWNER                 - owner of the work
     DIGEST                - digest value of the work
     PARTS                 - parts of the work, each of which is
                             a work itself
     CONTENTS              - indicator of where content of the work is
     COPIES                - number of copies of the work that are
specified
     COMMENT               - Comment
     SKU                   - Stock Keeping Unit, for extensibility to
allow
                             people to identify this work in their own
stock.
     RIGHTSGROUP           - rights group associated with the work
     REFERENCEDRIGHTSGROUP - reference rights group of the work
-->
<!ELEMENT WORK (OBJECT,
                DESCRIPTION?,
                CREATOR*,
                OWNER?,
                DIGEST*,
                PARTS?,
                CONTENTS?,
                COPIES?,
                COMMENT?,
                SKU?,
                ( RIGHTSGROUP |
                  REFERENCEDRIGHTSGROUP )+)
  >

<!-- Work Object -->
<!-- See OBJECT above -->

<!-- Work Description -->
<!ELEMENT DESCRIPTION (#PCDATA)>

<!-- Work Creator -->
<!ELEMENT CREATOR (#PCDATA)>
<!ATTLIST CREATOR
  type         CDATA      #IMPLIED
  >

<!-- Work Owner -->
<!ELEMENT OWNER ( %aPrincipal; )
  >
<!ATTLIST OWNER
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED
  >

<!-- Work Parts -->
<!ELEMENT PARTS (WORK+)>

<!-- Work Digest -->
<!-- See Digest above -->

<!-- Work Contents -->
<!-- If no CONTENTS is specified, the entire work represents content -->
<!ELEMENT CONTENTS EMPTY>
```

```
<!ATTLIST CONTENTS
  from          CDATA    #REQUIRED
  to            CDATA    #REQUIRED
  >

<!-- Work Copies -->
<!-- If no COPIES is specified, default is one
     If maxcount is not specified, then "unlimited" is assumed.
-->
<!ELEMENT COPIES EMPTY>
<!ATTLIST COPIES
  maxcount     CDATA   #IMPLIED
  >

<!-- Work Comment -->
<!ELEMENT COMMENT (#PCDATA)>

<!-- Work SKU (Stock Keeping Unit) -->
<!ELEMENT SKU (#PCDATA)>
<!ATTLIST SKU
  type         CDATA    #IMPLIED
  >


<!--================= Rights Group =================-->
<!-- a rights group for the current work
     COMMENT    - Comment on this rights group
     BUNDLE     - Bundled rights specification
     RIGHTSLIST - Individual rights specification
     name       - name of this rights group
-->
<!ELEMENT RIGHTSGROUP (COMMENT?,
                       BUNDLE?,
                       RIGHTSLIST)
  >
<!ATTLIST RIGHTSGROUP
  name    CDATA    #REQUIRED
  >

<!--================= Referenced Rights Group =================-->
<!-- A group of rights to be referred to by name in transport
     rights and applied to subsequent copies of the work.
-->
<!ELEMENT REFERENCEDRIGHTSGROUP (COMMENT?,
                                 BUNDLE?,
                                 RIGHTSLIST)
  >
<!ATTLIST REFERENCEDRIGHTSGROUP
  name    CDATA    #REQUIRED
  >


<!--================= BUNDLED RIGHTS =================-->
<!-- Terms and conditions applied to all the rights in the rights list -
->
<!ELEMENT BUNDLE (COMMENT?,
                  %termsConditions;,
                  WATERMARK?)
  >

<!--================= Rights List =================-->
```

```
<!-- Individual rights -->
<!ELEMENT RIGHTSLIST (COPY      |
                      TRANSFER  |
                      LOAN      |
                      PLAY      |
                      PRINT     |
                      VIEW      |
                      EXPORT    |
                      EDIT      |
                      EXTRACT   |
                      EMBED     |
                      BACKUP    |
                      RESTORE   |
                      VERIFY    |
                      FOLDER    |
                      DIRECTORY |
                      DELETE    |
                      INSTALL   |
                      UNINSTALL)+
  >

<!--===== Transport Rights =====-->

<!-- Copy Right -->
<!ELEMENT COPY (NEXTRIGHTS?,
                COMMENT?,
                (%termsConditions;)?)
  >

<!-- Transfer Right -->
<!ELEMENT TRANSFER (NEXTRIGHTS?,
                    COMMENT?,
                    (%termsConditions;)?)
  >

<!-- Loan Right -->
<!ELEMENT LOAN (REMAININGRIGHTS?,
                NEXTRIGHTS?,
                COMMENT?,
                (%termsConditions;)?)
  >

<!--===== Render Rights =====-->

<!-- Play Right -->
<!ELEMENT PLAY (PLAYER?,
                COMMENT?,
                (%termsConditions;)?,
                WATERMARK?)
  >

<!-- Print Right -->
<!ELEMENT PRINT (PRINTER?,
                 COMMENT?,
                 (%termsConditions;)?,
                 WATERMARK?)
  >
<!ATTLIST PRINT
    maxcount CDATA #IMPLIED
>
```

```
<!-- Viwer Right -->
<!ELEMENT VIEW (VIEWER?,
                COMMENT?,
                (%termsConditions;)?,
                WATERMARK?)
  >

<!-- Export Right -->
<!ELEMENT EXPORT (REPOSITORY?,
                  COMMENT?,
                  (%termsConditions;)?)
  >

<!--===== Derivative Work Rights =====-->

<!-- Edit Right -->
<!ELEMENT EDIT (EDITOR?,
                NEXTRIGHTS?,
                COMMENT?,
                (%termsConditions;)?)
  >

<!-- Extract Right -->
<!ELEMENT EXTRACT (EDITOR?,
                   NEXTRIGHTS?,
                   COMMENT?,
                   (%termsConditions;)?)
  >

<!-- Embed Right -->
<!ELEMENT EMBED (EDITOR?,
                 NEXTRIGHTS?,
                 COMMENT?,
                 (%termsConditions;)?)
  >

<!--===== File Management Rights =====-->

<!-- Backup Right -->
<!ELEMENT BACKUP (BACKUPRIGHTS,
                  COMMENT?,
                  (%termsConditions;)?)
  >

<!-- Right for backuped copy -->
<!ELEMENT BACKUPRIGHTS EMPTY>
<!ATTLIST BACKUPRIGHTS
  name    CDATA    #REQUIRED
  >

<!-- Restore Right -->
<!ELEMENT RESTORE (COMMENT?,
                   (%termsConditions;)?)
  >

<!-- Verify Right -->
<!ELEMENT VERIFY (VERIFIER,
                  COMMENT?,
                  (%termsConditions;)?)
  >
```

```
<!-- Folder Right -->
<!ELEMENT FOLDER (COMMENT?,
                  (%termsConditions;)?)
  >

<!-- Directory Right -->
<!ELEMENT DIRECTORY (COMMENT?,
                     (%termsConditions;)?)
  >

<!-- Delete Right -->
<!ELEMENT DELETE (COMMENT?,
                  (%termsConditions;)?)
  >

<!--===== Configuration Rights =====-->

<!-- Install Right -->
<!ELEMENT INSTALL (COMMENT?,
                   (%termsConditions;)?)
  >

<!-- Uninstall Right -->
<!ELEMENT UNINSTALL (COMMENT?,
                     (%termsConditions;)?)
  >

<!--===== Next Right =====-->
<!-- Right of the work after it has been transported or derived -->
<!ELEMENT NEXTRIGHTS (RIGHTSTOADD |
                      RIGHTSTODELETE)
  >

<!-- Right to Add -->
<!ELEMENT RIGHTSTOADD EMPTY>
<!ATTLIST RIGHTSTOADD
  name    CDATA    #REQUIRED
  >

<!-- Right to Delete -->
<!ELEMENT RIGHTSTODELETE EMPTY>
<!ATTLIST RIGHTSTODELETE
  name    CDATA    #REQUIRED
  >

<!-- Remaining Right -->
<!ELEMENT REMAININGRIGHTS EMPTY>
<!ATTLIST REMAININGRIGHTS
  name    CDATA    #REQUIRED
  >


<!--================= Named Principals in Rights =================-->

<!-- Player -->
<!ELEMENT PLAYER ( %aPrincipal; )
  >
<!ATTLIST PLAYER
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED
  >
```

```
<!-- Printer -->
<!ELEMENT PRINTER ( %aPrincipal; )
  >
<!ATTLIST PRINTER
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED
  >

<!-- Viewer -->
<!ELEMENT VIEWER ( %aPrincipal; )
  >
<!ATTLIST VIEWER
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED
  >

<!-- Repository -->
<!ELEMENT REPOSITORY ( %aPrincipal; )
  >
<!ATTLIST REPOSITORY
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED
  >

<!-- Editor -->
<!ELEMENT EDITOR ( %aPrincipal; )
  >
<!ATTLIST EDITOR
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED
  >

<!-- Verifier -->
<!ELEMENT VERIFIER ( %aPrincipal; )
  >
<!ATTLIST VERIFIER
  type         CDATA      #IMPLIED
  internal-id  CDATA      #IMPLIED
  >


<!--================= Terms and Conditions =================-->

<!--================= Time =================-->
<!ELEMENT TIME ( ( FROM     |
                   INTERVAL |
                   METERED)?,
                 UNTIL)
  >

<!-- From  -->
<!ELEMENT FROM ( %moment;)>

<!-- Interval  -->
<!ELEMENT INTERVAL EMPTY>
<!ATTLIST INTERVAL
  %interval;
  >

<!-- Metered -->
```

```
<!ELEMENT METERED EMPTY>
<!ATTLIST METERED
  %interval;
  >

<!-- Until -->
<!ELEMENT UNTIL ( %moment;)>


<!--================= Access Control =================-->
<!-- Access control conditions
     PRINCIPAL     - principal who accesses the work
     SECURITYLEVEL - security level requirement on the system
environment
                     in which the work is accessed
     SOURCE        - source principal if operation on the work involves
its
                     source
     DESTINATION   - destination principal if operation on the work
                     involves its destination
-->

<!ELEMENT ACCESS (PRINCIPAL*,
                  SECURITYLEVEL*,
                  SOURCE?,
                  DESTINATION?)
  >

<!-- Security Level -->
<!ELEMENT SECURITYLEVEL EMPTY>
<!ATTLIST SECURITYLEVEL
  name          CDATA    #IMPLIED
  value         CDATA    #REQUIRED
  >

<!--================= Named Principals fo Access Control
=================-->

<!-- A Source -->
<!ELEMENT SOURCE ( %aPrincipal; )
  >
<!ATTLIST SOURCE
  type         CDATA      #IMPLIED
  internal-id  CDATA            #IMPLIED
  >

<!-- A Destination -->
<!ELEMENT DESTINATION ( %aPrincipal; )
  >
<!ATTLIST DESTINATION
  type         CDATA      #IMPLIED
  internal-id  CDATA            #IMPLIED
  >

<!-- Certificate -->
<!ELEMENT CERTIFICATE (AUTHORITY,
                       PROPERTYPAIR*)
  >

<!-- Property Paire -->
<!ELEMENT PROPERTYPAIR EMPTY>
```

```
<!ATTLIST PROPERTYPAIR
  name          CDATA    #REQUIRED
  value         CDATA    #REQUIRED
  >


<!--================= Fee =================-->
<!-- form of payment
     TICKET   - to consume a ticket
     MONETARY - to bill a principal
-->
<!ELEMENT FEE (TICKET |
               MONETARY)
  >

<!-- Ticket -->
<!ELEMENT TICKET (AUTHORITY)
  >
<!ATTLIST TICKET
  type     CDATA    #REQUIRED
  id       CDATA    #REQUIRED
  >

<!-- Authority -->
<!ELEMENT AUTHORITY EMPTY>
<!ATTLIST AUTHORITY
  type     CDATA    #REQUIRED
  id       CDATA    #REQUIRED
  >

<!-- Money -->
<!ELEMENT MONETARY ( ( PERUSE         |
                       METEREDFEE     |
                       BESTPRICEUNDER |
                       CALLFORPRICE   |
                       MARKUP),
                     MIN?,
                     MAX?,
                     ACCOUNT )
  >

<!-- Per Use -->
<!ELEMENT PERUSE (CURRENCY?)>
<!ATTLIST PERUSE
  value         CDATA    #REQUIRED
  >

<!-- Currency -->
<!ELEMENT CURRENCY EMPTY>
<!ATTLIST CURRENCY
  iso-code      CDATA    #REQUIRED
  >

<!-- Metered Fee -->
<!ELEMENT METEREDFEE (RATE, PER, BY?)>

<!-- Metered Rate -->
<!ELEMENT RATE (CURRENCY?)>
<!ATTLIST RATE
  value         CDATA    #REQUIRED
  >
```

```
<!-- Metered Per -->
<!ELEMENT PER EMPTY>
<!ATTLIST PER
  %interval;
>

<!-- Metered By -->
<!ELEMENT BY EMPTY>
<!ATTLIST BY
  %interval;
  >

<!-- Best Price Under -->
<!ELEMENT BESTPRICEUNDER (CURRENCY?)>
<!ATTLIST BESTPRICEUNDER
  value         CDATA    #REQUIRED
  >

<!-- Call for Price -->
<!ELEMENT CALLFORPRICE EMPTY>
<!ATTLIST CALLFORPRICE
  dealer-id     CDATA    #REQUIRED
  >

<!-- Mark up -->
<!ELEMENT MARKUP EMPTY>
<!ATTLIST MARKUP
  percentage    CDATA    #REQUIRED
  >

<!-- Min and Max -->
<!ELEMENT MIN (RATE, PER)>
<!ELEMENT MAX (RATE, PER)>

<!-- Account -->
<!ELEMENT ACCOUNT ( ( ACCOUNTTO,
                      HOUSE? ) |
                    ( ACCOUNTFROM,
                      HOUSE? ) )
  >

<!-- Account To -->
<!ELEMENT ACCOUNTTO EMPTY>
<!ATTLIST ACCOUNTTO
  id            CDATA    #REQUIRED
  type          CDATA    #IMPLIED
  url           CDATA    #IMPLIED
  >

<!-- Account From -->
<!ELEMENT ACCOUNTFROM EMPTY>
<!ATTLIST ACCOUNTFROM
  id            CDATA    #REQUIRED
  type          CDATA    #IMPLIED
  url           CDATA    #IMPLIED
  >

<!-- House -->
<!ELEMENT HOUSE EMPTY>
<!ATTLIST HOUSE
```

```
  id            CDATA    #REQUIRED
  type          CDATA    #IMPLIED
  url           CDATA    #IMPLIED
  >


<!--================= Territory =================-->
<!-- physical location and digital domain at which a right can be
exercised
     LOCATION - physcial location
     DOMAIN   - digital domain
-->
<!ELEMENT TERRITORY ( (LOCATION |
                       DOMAIN)* )
  >

<!-- Location -->
<!-- location specifies an orgnization with its physical address -->
<!ELEMENT LOCATION (#PCDATA)>
<!ATTLIST LOCATION
  country       CDATA    #IMPLIED
  state         CDATA    #IMPLIED
  city          CDATA    #IMPLIED
  zipcode       CDATA    #IMPLIED
  street        CDATA    #IMPLIED
  >

<!-- Domain -->
<!-- domain specifies a system with its digital address -->
<!ELEMENT DOMAIN (#PCDATA)>
<!ATTLIST DOMAIN
  url       CDATA    #IMPLIED
  id        CDATA    #IMPLIED
  >


<!--================= Tracking =================-->
<!-- Information for tracking usage
     PROVIDERNAME - name of tracking principal
     PARAMETER    - parameter used for tracking
-->
<!ELEMENT TRACK (PRINCIPAL*,
                 PARAMETER*)
  >


<!--================= Watermark =================-->
<!-- Information for embedding watermarks -->
<!ELEMENT WATERMARK ( ( WATERMARK-STR |
                        WATERMARK-TOKENS |
                        WATERMARK-OBJECT)* )
  >

<!-- Watermark String -->
<!ELEMENT WATERMARK-STR EMPTY>
<!ATTLIST WATERMARK-STR
  string        CDATA    #REQUIRED
  >

<!-- Watermark Tokens -->
<!ELEMENT WATERMARK-TOKENS EMPTY>
```

```
<!ATTLIST WATERMARK-TOKENS
  all-rights           (true|false) "false"
  render-rights        (true|false) "false"
  user-name            (true|false) "false"
  user-id              (true|false) "false"
  user-location        (true|false) "false"
  institution-name     (true|false) "false"
  institution-id       (true|false) "false"
  institution-location (true|false) "false"
  render-name          (true|false) "false"
  render-id            (true|false) "false"
  render-location      (true|false) "false"
  render-time          (true|false) "false"
  copy-number          (true|false) "false"
  >

<!-- Watermark Object -->
<!ELEMENT WATERMARK-OBJECT (OBJECT)>
```

9 APPENDIX C. QUESTIONS AND ANSWERS

This section answers basic questions about usage rights specifications.

9.1 General Questions About Digital Property Rights

What kinds of computers can use digital works with digital property rights?

In principle, any computer can be made into a trusted system suitable to manage usage rights, at least for minimal levels of security. Different vendors will sell trusted systems with different levels of security. Most personal computers will be able to be trusted systems; other specialized devices (such as set-top boxes, personal audio equipment, and personal document readers) will eventually also be available as trusted systems.

Do publishers need to learn the rights markup language in order sell or distribute digital works?

No. Publishers need not learn the rights markup language any more than they need to learn Postscript. Publishers do need to know the different kinds of rights, conditions, and billing approaches that are appropriate for their The rights language is chiefly intended to be machine readable. Publishers would use "publishing systems" to import digital works into a trusted system and to assign rights using a graphical user interface. Typically, the interface would provide standard digital "boiler-plate" or defaults reflecting their typical sales approaches.

What's to keep somebody from just copying a digital file once they get it?

On a trusted system with a high security class, any attempt to copy a file without authorization will be prevented by the system software. Even on systems with low levels of security, where some protection mechanisms can be circumvented by non-trusted software, there are barriers to use of files (such as their encryption) that make them hard to use outside the control of a trusted system.

What's to keep people from just deleting the accounting information from their computers?

Accounting information is never kept in non-secure storage. On systems with low-level security, all transactions must be done when the computer is on-line - and all transactions are recorded at secure financial clearing house. Some trusted systems have secure storage, possibly in the form of removable security cards.

Even if computers have secure storage, what ensures that people will pay their bills?

Different people have different credit ratings and people do not like to damage their credit ratings. Some people will have repositories that work like credit cards, and others will have repositories that are prepaid debit cards. As with conventional credit cards, people who don't pay their bills will have trouble obtaining further credit.

Won't there be a lot of resistance by consumers to controlling use of digital works? (Look what happened to earlier "copy protection" schemes for software.)

Copy protection schemes have had two major flaws from a consumer point of view: (1) They do not offer any advantages to the consumer and (2) They get in the way of unconventional but fair uses of the material. The goal of copy protection schemes is

mainly to thwart unauthorized copying of digital works, without offering any advantages to consumers. The goal of usage rights is to promote a lively commerce in digital works. There are many possible advantages to consumers for digital publication. One advantage is the convenience of being able to order and accept immediate delivery of works any time of the day by phone, network, or even from a friend using consumer-based distribution. Another advantage is that it will often be possible and easy to arrange for inexpensive trial use of works for limited time periods. Finally, the cost of individual works to the consumer may often be lower if they choose to pay by the hour for works.

Will digital property rights create obstacles for consumers to use the works?

The goal in designing hardware and software for using digital works is to keep it simple. For example, on a trusted personal player, playing a digital work should be pretty much the way that playing is now -- you push the play button. When a user plays something the first time, he may need to insert his personal ID card or interact with an interface to choose whether his paying for the copy, paying by the use, or paying by the hour. For users of personal computers, the extra interactions with user interfaces will be minimal and use familiar styles.

What happens if somebody has an accident and loses a digital work. (For example, the repository may break or be lost?)

Different publishers may have different policies for handling these situations., depending on the price of the work and whether the work's ownership is known to the publisher. The usage rights approach has built-in provisions for governing such loss through the Backup and Restore rights.

What happens if a repository is compromised? (Who is responsible for the publisher's losses if a rogue copy gains wide circulation?)

Repository software and policies will be designed to manage the risks in the relationships among multiple platform vendors, financial clearing houses, publishers, and consumers. One approach is to have a certain low-level surcharge per transaction be used to cover insurance. The amount of risk can be reduced by a variety of other measures, including hot lists of compromised repositories and rogue works, digital certificates that expire, and access to relevant accounting data by law enforcement agencies.

What will it cost to update my Compaq 486 laptop computer to a good security level repository in September 1998?

Some questions are beyond the scope of this manual. Certainly the goal of this effort is to develop a market for affordable and secure enough solutions.

What happens when there is a conflict between usage rights specifications on a digital work and copyright law? For example, suppose that copyrights are supposed to expire after some period of time, but a digital work on a repository still insists on fees and conditions to use the work after that date?

This issue goes beyond the scope of this document. A casual analysis suggests that the precedence of copyright law and contract law (implied by usage rights) may be argued. The issue here is social and legal, not technological.

9.2 Questions About Work Specifications

Do all digital works have Work specifications?

Yes. A work specification is the top level specification for any digital work with usage rights.

How do I decide how to divide a work up into sections with separate work specifications?

Any part of a work that has distinct rights needs a separate work specification. Any part of a work that needs distinct permissions or distinct accounting of usage fees needs a separate work specification. When derivative works are made as a composition of separately created works, it is typical to have a separate work specification from each. If the works are made using the Extract and Embed rights, the work specifications would be made automatically.

Who assigns a work-id to a work?

The publisher assigns a work an identification using a publishing system. This is part of the process of importing a digital work into a repository. For the ID to be useful, the publisher would also want to make a reference copy available in a standard place, such as a public repository that they keep on-line or with an institution like the Library of Congress. The institutional arrangements for establishing reference copies have not been determined yet.

Who is the owner in a work specification?

The owner is the rights owner. Typically, this would be either the author or the publisher. The copy owner is the purchaser of a particular copy of a work.

How does a publisher get an authorization id?

An authorization id is something that comes with a digital certificate identifying the publisher. The usage rights approach assumes that there will be a small number of well-known authorities that issue certificates of identity. The public keys of the well known authorities would be known to essentially all trusted systems. We will refer to one such institution as the "Digital Property Trust" without being exact yet about its institutional nature.

Can someone publish a digital work without obtaining an ISBN or ISSN?

Yes.

What kinds of records do trusted systems keep about billing and usage?

Trusted systems keep track of all of the billable transactions on a work so as to carry out their accounting function.

Who would have access to usage data on trusted systems?

A basic principle is that these records should only be available to appropriate authorized institutions, and that institutions that access the information have certain obligations to protect privacy. Furthermore, authorized institutions need to have established and published policies about how they use the data.

9.3 Questions About Transport Rights

What is the difference between the Copy right and the Transfer right?

The Copy right and the Transfer right are both transport rights, in that they govern the flow of digital content between repositories. When the Copy right is exercised, a new copy is created on a second repository. After a Copy right has been exercised, the number of digital copies has been increased. Exercising a transfer right moves a copy from one repository to another. At the end of a transfer transaction, the original copy on the sending repository is deleted.

If someone has a work with a Copy right, does that mean that they can make digital copies and also print a document?

No. Before someone can exercise a Copy right, they must satisfy the fees and conditions associated with the right, which may include having particular authorization certificates (licenses) and paying certain fees. For example, the right might require that the receiving party have a certificate that they belong to a licensed organization or even that they old enough. Furthermore, the Copy right does not govern printing. It only governs the making of digital copies. The Print right governs the making of copies on external media, such as on printers or magnetic storage devices.

If I want to loan a digital work to a friend, can't I just exercise a Transfer right?

You could use a Transfer right, but you probably don't want to. Exercising a Transfer right removes the copy of the work from your repository. If your friend does not return the digital work, then the situation is much the same as when you loan somebody a paper book and never get it back. In contrast, the Loan right allows you to specify a loan period at the time of the loan. At the end of that period, the copy on your friend's repository deactivates and the copy on your repository reactivates, effecting an "automatic return."

If I borrow a copy of a digital work from a friend, can I make a copy of it?

This depends on the rights that are active on the loaner copy. A publisher uses the NEXTRIGHTS specification to indicate what rights are active. It is often in the publisher's interest to put Copy rights on loaner copies, because that enables more consumer-based sales. The right to make a copy of a loaned work is governed by the rights on the loaner copy. If there is a Copy right, then you can exercise that right. The Copy right specifies the fees and conditions that apply for making the copy.

If I loan a copy to a friend, can I still use it when it is loaned out?

Generally not. The usual arrangement is that only one person at a time can use a copy of a digital work. There are exceptions to this. For example, a user may have more than one copy of a work and can still use it as long as some copies are not yet loaned out. Also, a publisher can specify what rights remain behind during the period when a work is loaned out.

If no rights can be exercised when a digital work is loaned out, does that mean that the copy owner cannot even pay to get another copy?

Not necessarily. A publisher can use the REMAININGRIGHTS specification to indicate that certain rights can still be exercised when a work is loaned out. In particular, it is generally in the publisher's interest to have Copy rights remain on the work.

9.4 Questions About Render Rights

I can't find a Display right. What right would I use to read a copy of a work?

The Play right. The Play right is used generically to refer to ways for a user to experience a work. This includes displaying a work on a screen, playing a work over a speaker, playing a video game, and running a computer program.

I can play a digital movie on a screen, or digital music out a speaker. What is to keep someone from using a video camera or tape recorder to make a copy on magnetic tape?

Anything you can see or hear can probably be copied at some level of resolution on a video camera. Although unfair copying is generally prohibited by copyright law, usage rights and trusted systems can not block such copying. However, there are some deterrents. One deterrent is that generally the transition to and from analog signals introduces noise and degrades the quality of the copy. Also, it is possible to embed digital watermarks (or hidden signatures) in the work so as to make the copy traceable to the source. These could show up as almost unnoticeable marks on a page, sounds in music, or image alterations in video that could be used later for tracing the source.

If you can print out a work, what's to keep somebody from just making photocopies?

Copyright law governs the making of copies, but there is nothing in the technology for current generation copiers that stops a person from making photocopies of a paper document. Publishers can address this risk in several ways. First, they can limit who has the right to print a document. For example, they may limit this to people who are certified subscribers to the Copyright Clearance Center. Furthermore, they may require that printing take place on trusted printers that use watermarks. Watermarks make it possible to trace the user and repository from which the paper copy was printed.

9.5 Questions About Derivative Work Rights

How does a publisher know that fees and conditions will be honored when a published digital work is incorporated in somebody else's derivative work?

Derivative works are constructed by trusted systems whose function includes keeping the specifications for fees and conditions attached to the digital work. When derivative works are constructed in this way, the collection of fees and honoring of conditions is automatic and enforced by all of the repositories that are involved.

Would I need to obtain written permission to reuse a portion of a published digital work a new derivative work?

The reuse of a portion of a digital work is governed by the derivative work rights - Edit, Extract, and Embed. If a publisher has anticipated such uses and specified the rights and conditions, and you are agreeable to the rights and conditions, then you can just go ahead and reuse the work. The trusted systems will take care of billing and use restrictions automatically. This feature is intended to encourage the commercial reuse of works, providing a low overhead way for publishers to get compensation for such uses.

How could I say that a portion of a work can only be used for "professional" purposes? For example, I may not want certain digital pictures in a particular work to be used for entertainment publications.

Usage rights can only specify conditions that can be understood and enforced by trusted systems. One way to do this would be to require possession of a "professional" license from someone to exercise the Embed right. Another measure would be to require a "professional" license for someone to exercise a "Play" right.

9.6 Questions About File Management Rights

What's to keep me from making a backup copy, selling the original, and then exercising a Restore right?

Restore rights can have various conditions. For example, they may require tickets, fees, or communicating with the publisher's system. It is up to the publisher to choose conditions that adequately meet their security needs.

If I make a backup copy, where do I actually store it?

Backup copies of a file are actually encrypted files, with usage rights typically for restoring them, printing them, and transferring them. You could transfer the backup copy to another repository. You could also "print" them to external storage, such as an external disk for safe keeping. You also need to keep the file with the key for decrypting the backup file in secure storage on some trusted system.

Suppose that my friends and I commonly exchange digital works. How do I keep my friends from accessing my private files when we connect our repositories?

One way to do this is to organize your repositories into folders with different usage rights on different folders. Most repositories will come with a "private rights group" that requires your personal identity before rights can be exercised. By putting this rights group on your private folders, you can exclude others from using or even seeing what's in those folders.

If I have a digital work that I suspect is damaged or not genuine, how can I check it out?

Illegitimate copies of digital works will sometimes be circulated. These works might be created when someone manages to compromise a repository. They may also be created entirely from scratch, where someone copies a digital work manually and then offers it up for sale. Verifiers are systems that check the authenticity of a work. Some verifiers will be available on most repositories. These low-level verifiers can check encryption codes, hot-lists of rogue works, and so on. Other verifiers may compare a work to a reference copy. Such verifiers may be available on the network, where they have local access to the reference copy.

Do all works have Verify rights?

No, although almost all legitimate works will probably come with Verify rights with no associated fees or conditions.

Can I verify a work even if it does not have Verify rights?

Yes. In effect, an explicit Verify right makes it possible to transmit a copy of a work to a "verification server" without paying the required fees often associated with exercising a Copy right, and without losing a local copy of a work as in exercising a Transfer right. However, a work can always be transferred and subsequently returned. Some "players" and other programs can also perform verification functions.

If I delete a digital work, do I get my money back?

Probably not. Although it is possible for a publisher to give a credit when you delete a work, the more likely case is that deleting a work is like throwing away a book when you are done with it..

9.7 Questions About Time Specifications

Do all starting and expiration times on digital works need to be given in Greenwich Mean Time?

Yes and no. The times written in the language are always in terms of GMT. However, times are translated to and from the user's own time zone automatically by the user interface of the trusted system, so the user never actually sees the GMT times.

Do rights on works need to have expiration dates?

No. Expiration dates are optional. If no expiration date is given, then rights never expire.

What's the difference between a fixed interval and a sliding interval?

In a fixed interval, a right is valid from a fixed starting time to a fixed stopping time. In a sliding interval, the duration of the interval is fixed, but the interval starts the first time that the right is exercised.

What happens when there is a conflict between a metered interval and an expiration date? For example, suppose I have the right to use a work for a month, but the expiration date expires tomorrow.

In this example, the right expires tomorrow, even though there is potentially time left on the meter. If the publisher wanted to offer a month of use independent of an expiration date, he could specify that the rights never expire.

How will expiration dates effect libraries? Shouldn't libraries, or indeed anyone, be able to buy works without the rights expiring?

This is a good issue and reflects the social role of archiving of works. Authors and publishers that want to encourage archiving of their works should specify that the rights never expire.

Suppose that the right to use a digital work has not expired, but that the ticket a user has to exercise the work has expired. What should the user do?

Get a new ticket. Expiration dates on rights, certificates, and tickets are independent of each other. Everything that a user uses in a transaction must be valid at the time of the transaction.

On a personal computer, a user can sometimes fool the system by setting any time he wants. Why can't he do this on a repository?

Except for repositories of the lowest security classes, one of the requirements is that the clock be tamperproof. Furthermore, repositories check and compare their clocks at predetermined times in their protocols. For example, when repositories exchange works or register with financial clearing houses, clocks are compared.

Suppose that the batteries run down on a repository. How do you reset the clock?

Part of the function of a repository is to detect that a clock has stopped. Before any time-sensitive transactions can be carried out, the repository must obtain the time from another certified repository of adequate security level. Part of the communications protocol is to check and reset the clock under these conditions.

Suppose I buy a copy of a repository and then discover that all of the rights have expired?

Creating a work whose Copy rights expire long after other rights is a questionable publishing practice. Repository software can be made to check for such situations before money is exchanged to copy a work.

9.8 Questions About Fee Specifications

How do I get an account?

From an on-line bank. This could be a credit account or a debit account. The goal of this approach is to work within the initiatives already being developed for electronic commerce.

How does a publisher make digital tickets, and refer to them in rights?

Publishing systems have basic commands for making tickets and for including them in rights specifications. In this process, the publisher assigns expiration dates, usage rights on the tickets, identifiers, and so on. Digital tickets are digital works are distributed like other digital works and their use and transport is governed by usage rights.

Suppose that I pay to use a work by the hour, and then later decide to buy unlimited use of the work. Can I apply my metered fees towards the purchase?

The answer to this depends on the publisher. A repository treats these as two separate transactions. In principle, publishers can offer all sorts of rebates, specials, loyalty programs, and other incentives In some cases, a publisher would aggregate transactions according to the offer and then send credits to your billing account or grant you digital tickets that can be used with the publisher's products.

What is a "per-use" payment, that is, how do you define a single use of a work? What happens if a user starts using a work, is interrupted, and then starts again. Is that two uses? Alternatively, if a user plays a piece of music for two days, is that a single use?

In general, a per-use fee is charged every time that a right is exercised. For example, a per-use fee for a piece of music might be assessed every time the Play button is pressed. In the example given, where a user is interrupted and restarts the music, two uses would be counted. Similarly, if the music is played continuously for 2 days (using pause and reset or something), that would be a single play.
However, both of these examples can be modified by including a min-price-spec and a max-price-spec in the fee language. For example, the following right assesses a fee of fifty cents per use, but also caps the fee at fifty cents per day and requires a minimum fee of fifty cents per day. With this specification, the user can exercise the Play right as often as desired in a single day without paying more than fifty cents. Furthermore, if the user extends the Play for more than one day, he will pay an additional fifty cents.

```
<PLAY><FEE><MONETARY>
   <PERUSE value=".50"/>
   <MAX><RATE value=".50"></RATE><PER days="1"/>
    <MIN><RATE value="50"></RATE><PER days="1"/>
      <ACCOUNT>
         <ACCOUNTTO="Account-Jones-Pub248afdoiu4398"/>
   </ACCOUNT>
</MONETARY></FEE></PLAY>
```

In a Metered fee specification, is there a difference between a fee of sixty cents per hour and a fee of a penny per minute?

These rates can be the same or different, depending on other parts of the fee specification. The key issue is how the user is charged for partial use of a time period.

```
<PLAY><FEE><MONETARY>
   <Metered><RATE value=".60"> </RATE><PER hours="1"/><BY
minutes="1"/>
      <ACCOUNT><ACCOUNTTO="Account Jones-Pub-
245lkj5987"/> </ACCOUNT>
</MONETARY></FEE></PLAY>

<PLAY><FEE><MONETARY>
   <Metered><RATE value=".60"> </RATE><PER minutes="1"/>
      <ACCOUNT>
         <ACCOUNTTO="Account Jones-Pub-245lkj5987"/>
      </ACCOUNT>
</MONETARY></FEE></PLAY>
```

In the two play examples given here, both fees are computed the same way —to the nearest minute. The first example specifies the rate by the hour, but the "By" specification says that it is computed proportionately to the nearest minute of use. The

second example gives the rate by the minute. Since it does not have a By specification, it is computed to the nearest minute by default. In this way the rights markup language gives a publisher the flexibility to specify a rate in the most understandable terms.

In a Metered fee specification, is there a way to limit the maximum amount that a user needs to pay to exercise a right in a day?

Yes. This is another application of the maximum price specification. For example, the following specification charges sixty cents per hour accounted to the nearest minute but sets a maximum fee of two dollars per day.

```
<PLAY><FEE><MONETARY>
   <Metered><RATE value=".60"> </RATE><PER hours="1"/><BY
minutes="1"/>
   <MAX><RATE value="2.00"></RATE><PER days="1"/>
   <ACCOUNT><ACCOUNTTO="Account Jones-Pub-245lkj5987"/>
</ACCOUNT>
</MONETARY></FEE></PLAY>
```

What are digital tickets good for?

Digital tickets can be thought of as publisher-specific coupons. Publishers can distribute them to favored customers, or even bundle them with digital works. In general, they are used in loyalty programs to encourage consumers to go to a given publisher.

How can a publisher use digital tickets to give discounts on exercising a right?

Here are two examples of how digital tickets can be used in flexible pricing. In the first example, exercising the Play right costs a flat dollar per use.

```
<PLAY>
   <FEE><MONETARY>
         <PERUSE value="1.00"/>
   </MONETARY></FEE>
</PLAY>
<PLAY>
   <FEE><TICKET>
         <AUTHORITY id="Jones Publishers"/>
         <Type ticket-id=" Introductory Offer"/>
   </TICKET></FEE>
</PLAY>
```

In the second example, the Play right can be exercised at no fee by presenting a ticket. The ticket could be purchased ahead of time by the consumer. Its pricing and distribution is separate from the work and pricing may vary with time. The ticket could also be free, distributed as part of a sales promotion.

How does a best price specification work, since an off-line repository cannot know the best price at the time of the transaction?

A best-price specification indicates a maximum fee to be paid. This is the amount that is charged to the user at the time of the transaction. Later, after transactions are cleared at a financial clearing house, the publisher can give a rebate if a better price was in effect, crediting the user's account with the rebate amount. If a repository is on-line with the publisher at the time of the transaction, the actual price can be known and used in the transaction.

What happens if the publisher's price has gone up, above the best price maximum set in the usage right?

In this case, the user gets the lower price, since the publisher has already agreed to that rate. If a publisher expects a price to go up or has uncertainty about this, then the offer can be limited to a period of time by the time specification for the right.

How can a user exercise a call-for-price option if he is off-line?

He can't. The call-for-price specification requires a communication between the user and the rights owner at the time of the transaction.

What is a markup fee used for?

Markup fees are used to tack an additional percentage-based fee s for using a work. This could be used by distributors as a way of recovering costs. It could also be used as an automatic way to collect sales taxes.

Suppose that one publisher sells a digital work to a second publisher. Who gets the income stream from the usage of copies made and distributed by the first publisher?

When a digital work is sold, the contracts between the publishers could specify any way of dividing funds that makes sense. In this example, the works in the field would already have accounting information associated with the work. The accounting for dividing the continuing funds would need to be done by an accounting service trusted by both parties. For example, a financial clearing house could disburse the funds according to transactions. One way to anticipate this scenario would be for a publisher to obtain a unique account number for each digital work that it publishes. Funds accruing to such accounts could then be routed as desired.

9.9 Questions About Security and Authorization

What is the difference between a source authorization, a destination authorization, and a user authorization?

All authorizations are digital certificates. A source authorization is an authorization kept on the same machine as the digital work. It is usable in transactions by anyone accessing the work. A user authorization is associated with a particular individual. It is accessible to that individual when he is logged on to the repository. A destination authorization is an authorization that must be on a machine other than the one where the work resides.

Of what use is a destination authorization?

Destination authorizations are authorizations that must be on a repository other than the repository with the digital work during a transaction. They would appear in a right like the following:

```
<TREANSFER><ACCESS>
   <DESTINATION><CERTIFICATE>
       <AUTHORITY id="Jones Publishers"/>
       <PROPERTYPAIR name="License" value="ABC-Site-
poiuqer98743"/>
    </CERTIFICATE></DESTINATION>
 </ACCESS></TREANSFER>
```

In this example, a digital work cannot be transferred to another repository unless that repository has a particular license on it., no matter who is requesting the transfer.

How does a site license work in this approach?

A site license is a license for all people in a given organization to use a work. There are important variations on site licenses. What the approaches have in common is that a work has a right requiring a license to exercise it.

```
<PLAY><ACCESS>
   <PRINCIPAL type="User"><CERTIFICATE>
       <AUTHORITY id="Jones Publishers"/>
       <PROPERTYPAIR name="Type" value="Site-License"/>
    </CERTIFICATE></PRINCIPAL>
 </ACCESS></PLAY>
<PLAY><ACCESS>
   <SOURCE><CERTIFICATE>
       <AUTHORITY id="Jones Publishers"/>
          <PROPERTYPAIR name="Type" value="Site-License"/>
    </CERTIFICATE></SOURCE>
 </ACCESS></PLAY>
```

Here are two examples of how the authorization might be specified in a right. In the first example, the authorization is a user authorization. Such authorizations would typically be carried on a personal card. When a person uses a computer, he inserts the card and the system can access the license there. In the second example, the authorization is a "source" authorization, meaning that the license is on the computer itself. Anyone using the computer could exercise the right. The first version controls who the person is, and the second version controls which machine has the license.
Sometimes the license itself would require a separate and remote license server. For example, a site license may limit the number of simultaneous users of a work. In this case, the license on each machine would ask its local license server to contact a

centralized license server which counts the number of simultaneous users and admits new uses as long as the total count is less than some authorized number.

How would membership to a digitally-distributed journal work using this approach?

There are many different ways that on-line journals can be organized, so there is no single answer to this question.

A publisher could bundle printed distribution with on-line distribution, or could offer these as separately-priced options. A subscriber would get a subscription license, and potentially a number of subscription tickets useful for various things. For example, there could be on-line search services or print rights associated with digital tickets. Publishers could offer on-line bulletin boards and e-mail forums, open to subscribers. Some institutions could have site licenses.

How can a publisher prevent a digital work from being moved to a repository with low security?

A publisher can control the transport of a work by specifying security classes and other conditions on all transport rights. The security level of a platform is known to any platform that engages in a transaction with it. By specifying a high enough security class on all Copy, Loan, and Transfer rights, a publisher can preclude transport of the work to low security platforms. Furthermore, requirements for other digital licenses associated either with the person or physically secured repositories can also be specified.

What precludes authorizations from being forged or copied?

Authorizations are all digital certificates. To create an authorization requires knowing the private key of the authorizing agency. This makes them difficult to forge. The testing process checks one-way hash functions on the content. This makes it possible to detect tampering with the content. The transport of the certificates, like all digital works, is controlled by usage rights. Thus, these certificates are always kept in storage having high enough security. Even on semi-trusted systems, the certificates are kept in trusted storage. This makes certificates difficult to steal.

10 APPENDIX D. REFERENCES

[DPRL]

"The Digital Property Rights Language: Manual and Tutorial - XML Edition", Version 2.0, November 13, 1998. Xerox Corporation. It is available at http://www.contentguard.com.

[HTML]

"HTML 4.01 Specification", D. Raggett, A. Le Hors, I. Jacobs, 24 December 1999. Latest version available at: http://www.w3.org/TR/html401

[XML]

"Extensible Markup Language (XML) 1.0 Specification", T. Bray, J. Paoli, C. M. Sperberg-McQueen, 10 February 1998. Latest version available at: http://www.w3.org/TR/REC-xml

[XMLNAMES]

"Namespaces in XML", T. Bray, D. Hollander, A. Layman, 14 January 1999. XML namespaces provide a simple method for qualifying names used in XML documents by associating them with namespaces identified by URI. Latest version available at: http://www.w3.org/TR/REC-xml-names

[XMLSIGN]

"XML-Signature Syntax and Processing", Mark Bartel, John Boyer, Barb Fox, and Ed Simon, 28 February 2000. XML-Signature specifies XML digital signature processing rules and syntax. XML Signatures provide integrity, message authentication, and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere. Latest version available at: http://www.w3c.org/TR/xmldsig-core.

#

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P