CONTENTGUARD™

eXtensible rights Markup Language (XrML) 2.0 Specification
Part I: Primer

20 November 2001

Available formats: Microsoft Word and HTML. In case of a discrepancy, the HTML is considered definitive.

NOTE: To enable interactive browsing of the XrML schemas and examples, the XrML Specification and its companion Example Use Cases document use an HTML version that leverages the XML access functionality provided by the W3C Xpath recommendation. For this reason, you need to view these HTML documents with a browser that supports that recommendation (for example, Internet Explorer Version 6.0). If your browser does not support this functionality, please view the Microsoft Word versions of those documents.

Abstract

This specification defines the eXtensible rights Markup Language (XrML), a general-purpose language in XML used to describe the rights and conditions for using digital resources.

Status of this Document

Feedback and suggestions are welcome. Public discussion on XrML and its applications takes place on the discussion forum at http://www.xrml.org/forum.asp. Please report errors and provide comments on this document to the current editor at http://www.xrml.org/xrml_comments.asp.

Quick Table of Contents

Part I: Primer

- 1 About XrML
- 2 XrML Concepts
- 3 Extensibility of the XrML Core
- 4 Conformance

Part II: XrML Core Schema

- 5 Technical Reference

Part III: Standard Extension Schema

- 6 Standard Extensions

Part IV: Content Extension Schema

- 7 About the Content Extension
- 8 Content Extension Data Model
- 9 Content Extension Concepts

Part V: Appendices

- A XrML Schemas
- B Glossary
- C Index of Types and Attributes
- D References
- E Acknowledgements

Full Table of Contents for Part I: Primer

Preface 5

Content of this Specification 5

Scope of this Document 5

Dependencies on Other Specifications 5

Diagram Conventions 6

1 About XrML 7

1.1 The Need for a Language 7

1.2 Requirements for a Language that Describes Rights and Conditions 7

1.3 Benefits of XrML 7

1.4 Design Goals and Principles 8

2 XrML Concepts 9

2.1 License 10

2.2 Grant 11

2.3 Principal 12

2.4 Right 13

2.5 Resource 14

2.6 Condition 15

3 Extensibility of the XrML Core 16

3.1 Common XML Schema Extensibility Mechanisms 16

3.1.1 XML Schema Element Substitution Groups 16

3.1.2 XML Schema Type Substitution 16

3.1.3 XML Schema "any" Element 17

3.1.4 Using the Three Extension Mechanisms 17

3.2 Documenting Schema Extensions 18

4 Conformance 19

4.1 Markup Conformance 19

4.2 Application Conformance 19

Preface

Content of this Specification

This specification defines the eXtensible rights Markup Language (XrML), a general-purpose language in XML used to describe the rights and conditions for using digital resources. It also provides mechanisms to ensure message integrity and entity authentication within XrML documents. The specification consists of the following parts:

- Part I: Primer: Provides non-normative information about XrML including an overview of the language, an introduction to basic XrML concepts, and an explanation of the extensibility mechanisms the language provides.
- Part II: XrML Core Schema: Provides normative technical details regarding the core of the XrML design and architecture.
- Part III: Standard Extension Schema: Provides normative technical details regarding the XrML standard extension. This extension to the language defines types and elements common to many XrML usage scenarios but which do not form part of the language's core.
- Part IV: Content Extension Schema: Provides normative technical details regarding the XrML content extension. This extension to the language defines types and elements to describe rights, conditions, and metadata for digital works, allowing trusted systems to exchange digital works and interoperate.
- Part V: Appendices: Provides an index, a glossary, a list of references, and a list of acknowledgements.

Scope of this Document

This document explains the basic concepts for managing digital resources in trusted systems, describes the language syntax and semantics, and provides examples of typical specifications of rights and conditions. 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 industry to stipulate rights to use digital resources and the conditions under which those rights may be exercised. This document does not address the agreements, coordination or institutional challenges involved in achieving that goal.

Dependencies on Other Specifications

This specification depends on the following other specifications:

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

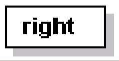
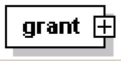
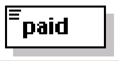
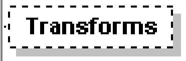
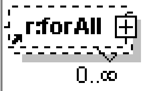
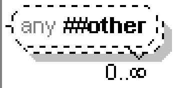
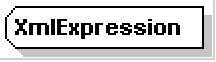
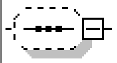
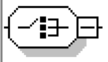
Namespaces in XML
T. Bray, D. Hollander, A. Layman, 14 January 1999.
http://www.w3.org/TR/REC-xml-names

XML Schema
David C. Fallside, Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, Paul V. Biron, Ashok Malhotra, 02 May 2001
http://www.w3.org/XML/Schema

For a list of other related specifications, see Appendix D, References.

Diagram Conventions

The diagrams in this specification use the following conventions:

	A single mandatory element.
	A single mandatory element with child elements.
	A single mandatory element containing Parsed Character Data (#PC-Data). The content may be simple content or mixed complex content.
	A single optional element.
	A multiple optional element (in this case, zero to infinity) with child elements.
	A placeholder for any element from any namespace.
	A complex type.
	A sequence. Solid lines connect this symbol to required elements within the sequence. Dotted lines connect this symbol to optional elements in the sequence.
	A choice.

Note: These diagrams were generated using Altova XML Spy Integrated Development Environment. For more information, see the Altova web site, http://www.xmlspy.com/.

1 About XrML

This chapter provides an overview of XrML. It provides a basic definition of XrML, describes the need that XrML is meant to address, and explains design goals for the language.

1.1 The Need for a Language

The reality of the Internet and the need to control the use of digital content and digital services has fueled the development of technologies that attempt to manage, secure, control, and automate the flow of content and the access of services. Digital Rights Management (DRM) is the common term associated with such technologies. Other technologies such as Digital Asset Management, Content Management, and Trust Systems are also getting incorporated into the DRM workflow. The DRM space is becoming more important and, in many cases, required to enable certain business models.

If we consider the lifecycle or workflow for digital content and services, we see that exchange of *rights information* is required between the players or entities in the workflow or at each step of the lifecycle. For example, a content user needs to know what rights or permissions are associated with a piece of content.

We also realize that expressing rights can be simple or very complex. For example:

- A user may obtain the rights for *unlimited play* for a music file.
- A corporate document may have usage right restricted to certain managerial levels, PCs, and/or certain dates.

Rights expressions get more complex when we try to mimic the use and distribution of content in the physical world, for example, when specifying the rights that would govern lending of a digital book, or the rights that control the giving away of an article in an electronic magazine.

Therefore, a common language that can be shared among the participants in this digital workflow is required, not only from an obvious interoperability point of view, but more so to comprehend that rights will be manipulated and changed during the digital workflow and lifecycle. For example, the usage rights for a piece of content will change as it moves among the creator, aggregator, retailer, and user.

1.2 Requirements for a Language that Describes Rights and Conditions

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

- *Comprehensive*: A language shall be capable of expressing simple and complex rights expressions in any stage in a workflow, lifecycle or business model.
- *Generic*: A language shall be capable of describing rights for any type of digital content or service (an ebook, a file system, a video, or a piece of software).
- *Precise*: a language shall communicate precise meaning to all the players in the system.

1.3 Benefits of XrML

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

XrML has its roots in Xerox PARC's Digital Property Rights Language (DPRL), first introduced in 1996. DPRL became XrML when the meta-language (used to construct the language) changed from lisp to XML in 1999. Since then, ContentGuard and its partners have invested additional work to enrich the language.

XrML has thus become *comprehensive* by providing a framework to express rights at different stages of a workflow or lifecycle, *generic* by defining a large body of format and business neutral terms (about 100) and using these terms to specify rights to any digital content and any digital services, and *precise* through the development of a grammar and processing rules that allow unique interpretation of the language. XrML is by far the most advanced and mature rights language.

Additionally, XrML exploits the advantages provided by current XML technologies:

- The XrML *syntax* is defined by XML.
- The XrML *grammar* is defined by XML Schema Definitions (xsd).
- Language extensibility leverages the rules defined by XML and XSD. This allows the introduction of new elements (rights terms) without breaking the language

This means that XrML can leverage XML technologies and know-how to machine-process the language, to extend the language to support new business models, and to integrate the language with other technologies and systems.

Furthermore, XrML comprehends real-life system issues such as:

- Ensuring the authenticity of the XrML rights specification through the use and application of digital signatures.
- Specifying the level of trust that is needed by the parties involved.
- Supporting the identification and discovery of resources through the use of UDDI.
- Supporting pattern matching through XPath.

1.4 Design Goals and Principles

A principal design goal for XrML 2.0 is to accommodate and support extensibility and customizability without changes to XrML 2.0 itself. To achieve this goal, the XrML 2.0 design follows the principles listed below:

- The syntax is described and defined using XML Schema, defined by the Worldwide Web Consortium (W3C). The syntax extensively exploits the XML Schema typing system. More powerful and expressive than DTD technology, use of XML Schema maximizes the richness and flexibility of the XrML 2.0 architecture.
- The design accommodates independent extension by third parties without compromising the typing infrastructure. XrML 2.0 can be extended in a number of ways, including definition of new and application-specific rights, resources, and conditions. Judicious use of XML Namespaces guarantees that extensions do not to conflict even though the parties that defined them have not coordinated their efforts or synchronized with revisions and updates to the XrML 2.0 core schema.
- The flat structure of grants is agnostic and even-handed about how grants may be bundled together. For example, it is equally straightforward and succinct to issue several grants to a single issued-to party as to issue several grants about a certain digital work to various different parties involving various different actions. This agnosticism is important to the pragmatic ease and attractiveness with which licenses can be used across the spectrum to which they are appropriate.
- XrML 2.0 provides a simple, orthogonal mechanism that avoids repetition of identical information within a license with minimal impact on the overall XML element structure.

Other design goals for XrML include:

- 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 leverage existing standards or standard activities as much as possible, including W3C XML schema, XML Digital Signature, and XPath.

XrML 2.0 itself makes use of its extensibility mechanisms internally. It is split into several parts:

- a core schema, containing definitions of concepts that are at the heart of the semantics of XrML 2.0, particularly those having to do with evaluation of a trust decision.
- a standard extension schema, containing definitions of concepts which are generally and broadly applicable to XrML 2.0 usage scenarios, but which are not at the heart of XrML 2.0 semantics.
- a content specific extension schema that defines rights management concepts specifically related to digital content or works such as books, music, and video.

Others parties may define their own extensions using existing, standard XML Schema and XML Namespace mechanisms.

2 XrML Concepts

This chapter describes the key concepts and the data model used by XrML to define rights and conditions for principals to use digital resources.

XrML 2.0 core concepts include license, grant, principal, right, resource and condition. The XrML Core defines elements for the last four of these concepts in an abstract fashion. Extensions to the XrML Core could extend these elements to address specific business applications.

A simple XrML license looks like the following, where the keyHolder is a principal, print is a right (defined in the XrML Content Extension), digitalWork is a resource (also defined in the XrML Content Extension), and validityInterval is a condition.

A particular key holder can print an ebook located at a given URL before Christmas of 2001.

```
<license>
   <grant>
      <keyHolder>
         <info>
            <dsig:KeyValue>
               <dsig:RSAKeyValue>
                  <dsig:Modulus>Fa7wo6NYfmvGqy4ACSWcNmuQfbejSZx7aCibIg
                  kYswUeTCrmS0h27GJrA15SS7TYZzSfaS0xR9lZdUEF0ThO4w==
                  </dsig:Modulus>
                  <dsig:Exponent>AQABAA==</dsig:Exponent>
               </dsig:RSAKeyValue>
            </dsig:KeyValue>
         </info>
      </keyHolder>
      <cx:print/>
      <cx:digitalWork>
```

```
          <cx:locator>
             <nonSecureIndirect URI="http://www.contentguard.com/sampleBook.spd"/>
          </cx:locator>
       </cx:digitalWork>
       <validityInterval>
          <notAfter>2001-12-24T23:59:59</notAfter>
       </validityInterval>
    </grant>
</license>
```

2.1 License

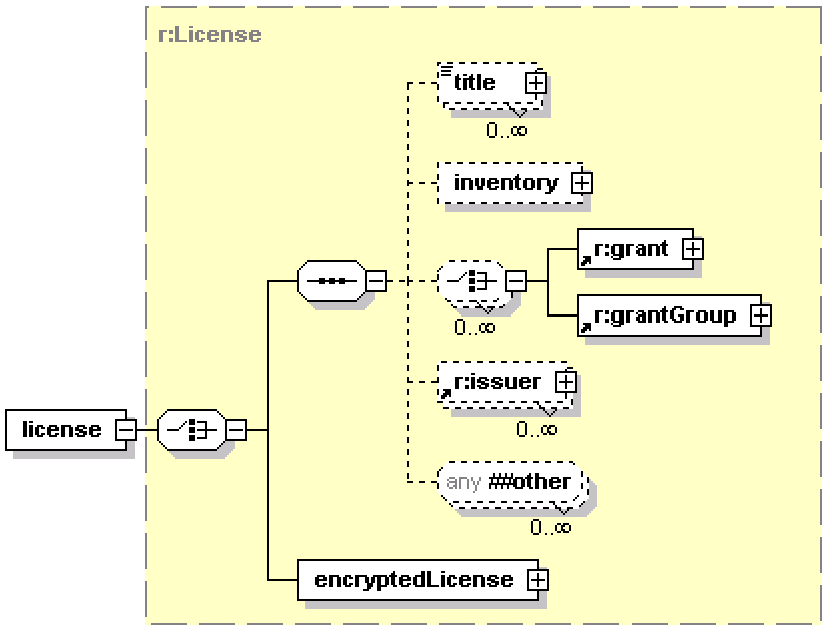
A key top-level construct in XrML 2.0 is a license. Conceptually, a license is the container of grants. The basic structure of a license contains the following:

- a set of grants that convey to certain principals certain rights to certain resources under certain conditions
- an identification of the principal or principals who issued the license and thus bestow the grants upon their recipients
- miscellaneous additional information

Those familiar with XrML 1.x, digital certificates, and other similar structures might notice the absence of the identification of the principal or principals to whom certain rights are conveyed (e.g., ISSUEDPRINCIPALS in XrML 1.x). This notion has been flattened and regularized within the structure and terms of each individual grant.

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

License Model



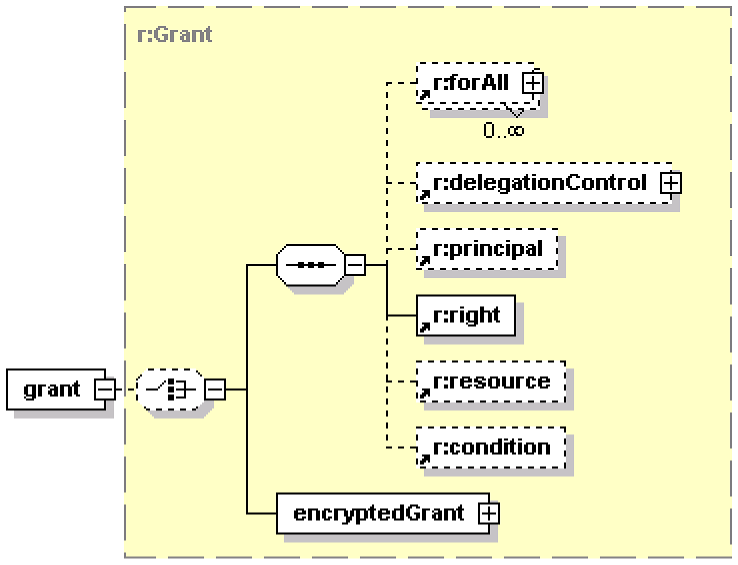
2.2 Grant

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

Structurally, a grant consists of the following:

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

Grant Model



The XrML 2.0 Core defines principals, rights, resources, and conditions as abstract concepts. It is expected that extensions to the *Principal*, *Right*, *Resource*, and *Condition* types will define useable principals, rights, resources, and conditions, respectively.

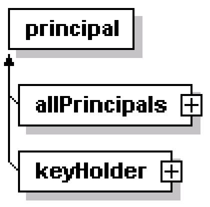
2.3 Principal

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

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

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

Principal Model

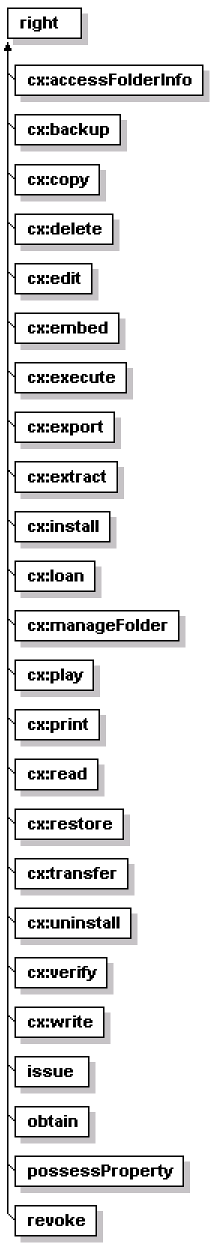


Extensions to the XrML Core could use the Principal element in any context in which a party must be identified and authenticated. For example, the Principal element could be used to identify a resource (for example, to identify and authenticate a secure service to which rights are granted).

2.4 Right

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

Right Model

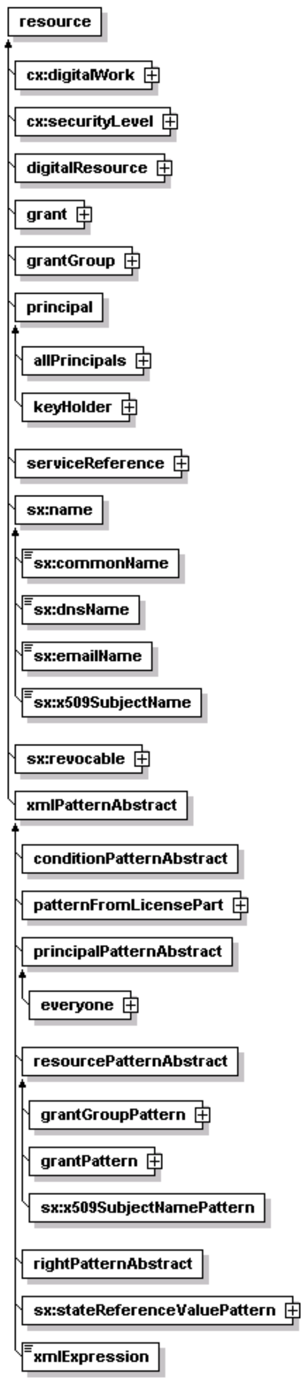


The XrML 2.0 Core provides a *right* element to encapsulate information about rights and provides a set of commonly-used, specific rights, notably rights relating to other rights, such as *issue*, *revoke*, *delegate*, and *obtain*. Extensions to the XrML Core could define rights appropriate to using specific types of resources. For instance, the XrML content extension defines rights appropriate to using digital works (for instance, *play* and *print* rights).

2.5 Resource

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

Resource Model

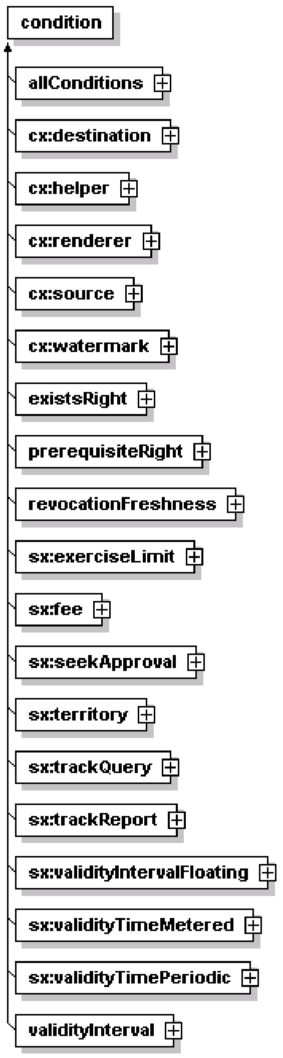


The XrML 2.0 Core provides mechanisms to encapsulate the information necessary to identify and use a particular resource or resources that match a certain pattern. The latter allows identification of a collection of resources with some common characteristics. Extensions to the XrML Core could define resources appropriate to specific business models.

2.6 Condition

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

Condition Model



The XrML 2.0 Core defines a *condition* element to encapsulate information about conditions and some very basic conditions. Extensions to the XrML Core could define conditions appropriate to specific distribution models. For instance, the XrML content extension defines conditions appropriate to using digital works (for instance, watermark, destination, and renderer).

3 Extensibility of the XrML Core

The XrML Core uses XML Schema technology to enable extensibility. For further details about these mechanisms, please refer to the XML Schema Primer.

3.1 Common XML Schema Extensibility Mechanisms

Three XML Schema technologies are used commonly with XrML for extensibility:

3.1.1 XML Schema Element Substitution Groups

To extend XrML using this mechanism, define a new element with a type derived from an existing element's type and set the "substitutionGroup" schema attribute of the new element to the name of the existing element. This type of extension is the most commonly used with XrML. It is the type of extension used by *licensePart*, *principal*, *right*, *resource*, *condition*, and *xmlPatternAbstract*.

Example: Creating a *timeLimit* condition

```
<xsd:element name="timeLimit" type="myext:TimeLimit"
substitutionGroup="r:condition"/>
   <xsd:complexType name="TimeLimit">
      <xsd:complexContent>
         <xsd:extension base="r:Condition">
            <xsd:sequence minOccurs="0">
               <xsd:element name="duration" type="xsd:duration"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
```

To use this type of extension, use the new element in the existing element's place.

3.1.2 XML Schema Type Substitution

To extend XrML using this mechanism, define a new type derived from an existing type.

Since this type of extension is similar to element substitution groups, many people choose to use element substitution groups instead. Both can be used in most situations. The primary difference is that with element substitution groups, the name of the element changes, whereas with type substitution, an xsi:type attribute is used instead.

Example: Adding *locationOfIssue* to *IssuerDetails*

```
   <xsd:complexType name="MyDetails">
      <xsd:complexContent>
         <xsd:extension base="r:IssuerDetails">
            <xsd:sequence>
               <xsd:element name="locationOfIssue" type="xsd:string">
               </xsd:element>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
```

To use this type of extension, place the xsi:type attribute in your instance document on the element whose type you are changing.

3.1.3 XML Schema "any" Element

The "any" element placeholder defines an extensibility point which may contain any XML element from any other namespace. To extend XrML using this feature, define any element in any other namespace.

XrML allows this extensibility mechanism at select points where the extension space can be large enough to account for a whole standard specification. Examples of this are extra license information, methods of referring to a service, and mechanisms for revocation.

Example: Adding proprietary secret information to a license

```
 <xsd:element name="proprietarySecretInfo" type="xsd:base64Binary"/>
```

To use this type of extension, use the element in the position defined by the "any" element placeholder.

3.1.4 Using the Three Extension Mechanisms

The following license gives an example of using these three extensions. The example grants anyone the right to execute a certain game (located at http://www.xrml.org/games/2001/11/myext) for up to 10 minutes. The details show that the license was signed in the United States. The proprietary secret information might contain the key to unlock the game on some secure game demonstration system.

Example: Using the three extension mechanisms

```
<license>
   <grant>
      <forAll varName="anyone"/>
      <principal varRef="anyone"/>
      <cx:execute/>
      <digitalResource>
         <secureIndirect URI="http://www.xrml.org/games/2001/11/myext">
            <dsig:DigestMethod
               Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <dsig:DigestValue>PB4QbKOQCo941tTExbj1/Q==</dsig:DigestValue>
         </secureIndirect>
      </digitalResource>
      <myext:timeLimit>
         <myext:duration>PT10M</myext:duration>
      </myext:timeLimit>
   </grant>
      <issuer>
      <dsig:Signature>
         <dsig:SignedInfo>
            <dsig:CanonicalizationMethod Algorithm=
               "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <dsig:SignatureMethod Algorithm=
               "http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <dsig:Reference>
               <dsig:Transforms>
                  <dsig:Transform Algorithm=
                     "http://www.xrml.org/schema/2001/11/xrml2core#license"/>
               </dsig:Transforms>
               <dsig:DigestMethod Algorithm=
                  "http://www.w3.org/2000/09/xmldsig#sha1"/>
               <dsig:DigestValue>PB4QbKOQCo941tTExbj1/Q==
               </dsig:DigestValue>
            </dsig:Reference>
         </dsig:SignedInfo>
         <dsig:SignatureValue>AYmqOhSHbiP9JadD2GLBweJdGzNNbwDgFDBtjpRn2aeW0MG
            XFF9zmSaN46kylPb7ZQAPozk8Cf5V5u9kQrk5QQ==
         </dsig:SignatureValue>
         <dsig:KeyInfo>
            <dsig:KeyValue>
               <dsig:RSAKeyValue>
                  <dsig:Modulus>X0j9q9OAxvhre4NP6qdF11LktdToQQmj7RZYPNnaIpg
                     OyM4c1T34rcnDTf84oRUTUiTQkMkne05g7cQ2W00yzA==
                  </dsig:Modulus>
                  <dsig:Exponent>AQABAA==</dsig:Exponent>
               </dsig:RSAKeyValue>
            </dsig:KeyValue>
         </dsig:KeyInfo>
      </dsig:Signature>
      <details xsi:type="myext:MyDetails">
         <timeOfIssue>2001-11-19T16:20:01</timeOfIssue>
         <myext:locationOfIssue>US</myext:locationOfIssue>
      </details>
   </issuer>
   <myext:proprietarySecretInfo>D2GLBweJdGzNNbwDgFDBtjpRn2aeW==
   </myext:proprietarySecretInfo>
</license>
```

3.2 Documenting Schema Extensions

When making extensions, it is important to clearly document the semantics of the new elements and types. The following table lists things to consider when documenting common classes of extensions:

Class of Extension	**Documentation Considerations**
principal	Describe what procedure should be followed before making a claim that a certain principal can be described by an instance of that extension.
right	Describe which sequences of events are consistent, and which are inconsistent, with the right described by an instance of that extension.
resource	Describe which physical or logical items are described by instances of that extension.
condition	Describe the set of contexts for which an instance of that extension is to be considered satisfied and the set of contexts for which an instance of that extension is to be considered breached.
xmlPatternAbstract	Describe the set of XML trees that match the pattern and the set that do not.

4 Conformance

This chapter provides conformance requirements and guidelines for XrML. These conformance requirements and guidelines ensure consistency of format, organization, content, interpretation, and other aspects of an XrML expression or construct, so that certain levels of reuse and interoperability can be achieved among XrML applications.

This chapter addresses conformance in terms of the following two aspects:

- *markup conformance*: refers to the conformance of any XML expressions that use valid XrML expressions. The "use" includes using only XrML expressions as well as using XrML expressions together with XML expressions within other namespaces. The conformance is on the syntactic and semantic information that the XML expressions carry.

- *application conformance*: refers to the conformance of any XML applications that are capable of processing XrML expressions. The conformance is on their functionality for performing syntactic validation and semantic interpretation of XrML expressions according to the XrML specification.

4.1 Markup Conformance

An XML expression is said to be XrML *markup conforming* if it contains elements in the XrML core and extension namespaces, and these elements together with their attributes meet all the mandatory syntactic and semantic requirements as prescribed in the normative parts of this XrML specification. An XrML markup conforming XML expression can be an XML expression starting with an XrML element or an XML expression that uses XrML elements together with elements in other XML namespaces. Since the XrML schemas are normative, XrML expressions must be valid against these schemas in the sense defined by XML Schema. Moreover, they must adhere to the semantic restrictions imposed for each XrML element and attribute used. For instance, a licensePart cannot have both the attributes licensePartId and licensePartIdRef, even though having both of them is syntactically valid.

4.2 Application Conformance

An XrML application is any application or software/hardware module that can validate, interpret, or generate valid XrML expressions. An XrML application that processes XrML expressions might be an editor that creates or modifies XrML expressions, or an interpreter that verifies, against an XrML license expression, whether or not some principal has some right on some resource.

An XrML application is said to be XrML *application conforming* if, when it claims to validate, interpret, and/or generate XrML, it validates the syntax of, interprets the semantics of, and/or generates well-formed XrML expressions according to the XrML specification. An XrML conforming application must meet the following criteria:

- It implements and processes the mandatory elements and attributes as well as mandatory semantic requirements ("must" and "must not") set forth in this specification.

- For any optional elements and attributes as well as optional semantic requirements ("should" and "may") it chooses to implement, it implements and processes them in the way prescribed.

- At a minimum, it implements the mandatory elements and attributes in the XrML Core.

- If it implements any XrML extension, it must implement all the mandatory elements and attributes in that extension.

- It must parse and check an XrML document for well-formedness. If the application has validation functionality, it must also validate XrML documents against their referenced schemas and the semantic requirements specified in this XrML specification.

- When it has interpretation functionality, it must interpret XrML in ways consistent with the semantic definitions and processing rules specified in the XrML specification. This covers processing rules for handling licensePartId and licensePartIdRef, pattern matching, using varName and varRef, generating and verifying license signatures, and authorization.

Go to Part II: XrML Core Schema