CONTENTGUARD™

eXtensible rights Markup Language (XrML) 2.0 Specification
Part IV: Content Extension Schema

20 November 2001

Available formats: HTML and PDF. 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 PDF versions of those documents.

Quick Table of Contents

Full Table of Contents for Part IV: Content Extension Schema

7 About the Content Extension

The XrML content extension is an extension to XrML 2.0 that describes rights, conditions, and metadata for digital works, allowing trusted systems to exchange digital works and interoperate.

Trusted systems (or repositories) are systems that can hold digital works and that can be trusted to honor the rights and conditions specified for them. For example, in document commerce, trusted systems are for authoring, playing, and selling digital works. They include personal systems, on-line storefront systems, library systems, and so on.

Design goals for the XrML content extension are:

- To enable content owners and distributors to describe rights 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 the language features to meet the needs of the digital content industry today and as it develops in the future.

The XrML content extension is defined using XML Schema and extensively exploits its typing system.

8 Content Extension Data Model

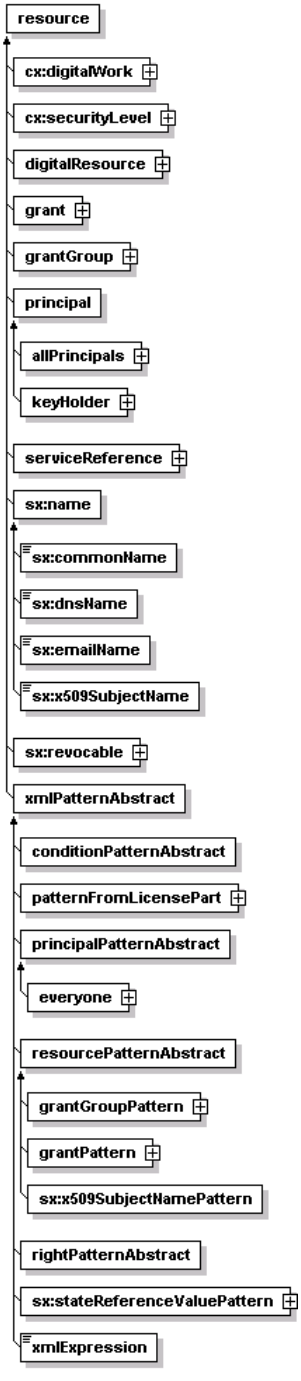
This chapter describes the basic elements for the XrML content extension. These elements describe following:

- The Resources to which Rights and Conditions apply. The XrML content extension defines resources that encapsulate DigitalWorks and their metadata.
- Rights to distribute or use a DigitalWork.
- Conditions under which Rights may be exercised.

8.1 Resources

The XrML content extension extends the resource element defined by the XrML 2.0 Core to address digital works. Specifically, the XrML content extension defines a DigitalWork type that encapsulates information about a digital work.

Resource Model

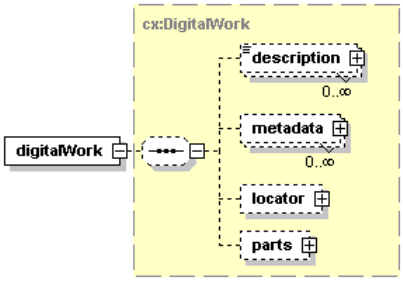


8.1.1 The Digital Work

The DigitalWork represents the digital content to which Rights and Conditions are being applied. The digital work consists of the following:

- A description of the work, which may be provided in several languages.
- metadata for the work (see below).
- A locator for the work's content.
- The parts of this work, each of which is itself a DigitalWork.

Digital Work Model



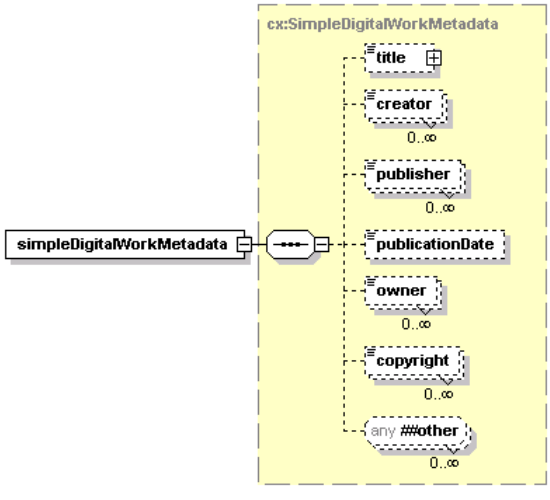
8.1.2 Metadata

metadata specifies information about the DigitalWork. This information may be used in many contexts. For example, an eCommerce web site may use this information to construct a catalog of works for sale.

The SimpleDigitalWorkMetadata provides the following information:

- The title of the work.
- The creators of the work, which may or may not be the same as the owners.
- The publishers of the work, which may or may not be the same as the owners.
- The publicationDate (and optionally the time on that date).
- The owners of the work, which may or may not be the same as the creators or publishers. The owners hold copyright on the work.
- The formal copyright declarations for the work.
- Any other information for the work.

Simple Digital Work Metadata Model



8.2 Rights

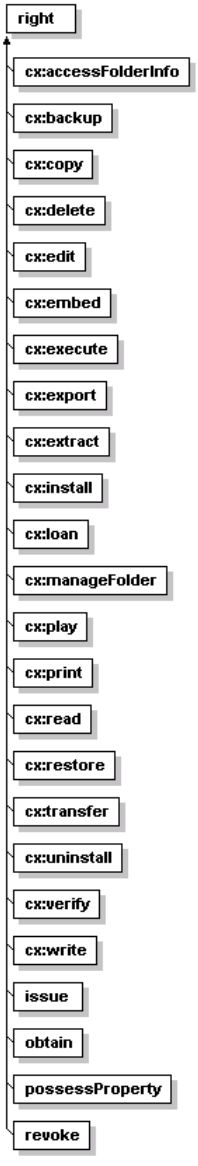
When using the XrML content extension, Rights are associated with a DigitalWork to describe how it may be distributed and/or used. Each Right has a corresponding transaction that defines what a trusted system or repository does when that right is exercised.

The XrML 2.0 Core and Standard Extension define some general-use rights. The XrML content extension defines Rights specific to distributing and using DigitalWorks. Conceptually, these Rights are grouped by type as follows:

Render Rights	Govern the rendering of a DigitalWork. The XrML content extension defines the following render Rights: Play: Render the work in a transient form, as appropriate to the content type. Depending on the content type, exercising the Play

	`Right` might result in displaying a book, playing an audio clip, or showing a video. `Print`: Make a permanent non-digital rendered copy of the work outside the control of a repository. Exercising the `Print` `Right` might result in a printing a hard copy of a book or creating an audio recording on a magnetic tape. `Export`: Make a digital source copy of the work without any `Right`s or `Condition`s associated with it.
Transport Rights	Govern the movement of a `DigitalWork` from one repository to another. The XrML content extension defines the following transport `Right`s: `Copy`: Make a copy of the work. `Transfer`: Transfer the work to another repository, removing the protected content from the original location. `Loan`: Lend the work for a specific period of time. While the work is on loan, the original copy cannot be used.
Derivative Work Rights	Govern the reuse of a digital work, in whole or in part, to create a new or composite work. `Edit`: Make changes to work to create a new work based on the original. `Extract`: Use a portion of the work to create a new work. `Embed`: Include the work as part of a composite work. An `Embed` operation places a copy of the work inside the composite work.
File Management Rights	Govern two types of operations: • Access to the directory information between repositories. These `Right`s are needed whenever two repositories need to communicate, such as when `Transfer` or `Loan` `Right`s are exercised. • Make and restore backup copies of a `DigitalWork`. `Read`: Read the work from the repository. `Write`: Write the work in the repository. `Execute`: Execute the work from the repository. `Delete`: Delete the work from the repository. `Backup`: Create a backup copy of the work. `Restore`: Restore a digital work from a backup copy. `Verify`: Check the authenticity of the work in the repository. `ManageFolder`: Create and name subfolders and configure folders by moving files and subfolders among them. `AccessFolderInfo`: Obtain information about the work within folders in a repository.
Configuration Rights	Govern the addition and removal of system software from a repository. `Install`: Make software runable on the repository, including checking that the software is certified, that it has not been tampered with, and that it is compatible with the repository. `Uninstall`: Disable software from running, restoring it to the state it was in before it was installed. Exercising an uninstall `Right` does not remove the files for the program from the repository.

Right Model



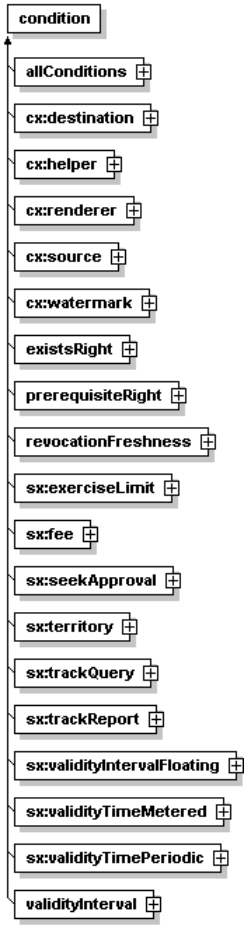
8.3 Conditions

Conditions specify the terms, conditions, restrictions, and other modifiers applied to the exercise of `Right`s. These elements can be used to specify, for instance, watermarks to be applied when using a work or other restrictions on use.

The XrML 2.0 Core and Standard Extension define some general-use `Condition`s, such as fees and expiration times. The XrML content extension defines the following `Condition`s specific to distributing and using `DigitalWork`s:

`Destination`	Limits the repositories to which a work can be moved.
`Helper`	Limits the software that can be used to exercise a `Right`.
`Renderer`	Identifies the device that can be used to render a work.
`Source`	Limits the source device to use when exercising a `Right`.
`Watermark`	Specifies a digital watermark.

Condition Model



9. Content Extension Concepts

The XrML Content Extension extends the XrML Core by defining rights, resources and metadata, and conditions and obligations related to digital content management.

9.1 Content Extension Rights

This section describes each of the `Right` elements defined by the XrML Content Extension.

Each XrML Content Extension `Right` element has a corresponding type that extends the `Right` complex type defined in the XrML Core.

9.1.1 The AccessFolderInfo Right

Represents the right to obtain information about the works within folders in a repository.

Schema Representation of the `AccessFolderInfo` Type

```
-<xsd:complexType name="AccessFolderInfo">
 -<xsd:complexContent>
    <xsd:extension base="r:Right"/>
  </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `accessFolderInfo` Element

```
-<xsd:element name="accessFolderInfo" type="cx:AccessFolderInfo" substitutionGroup="r:right">
 </xsd:element>
```

Using the `AccessFolderInfo` `Right`

The `AccessFolderInfo` type represents the `Right` to deliver or reveal information about the works contained within folders.

`AccessFolderInfo` is one of the file management rights.

Alice has the `Right` to get a directory listing of the Specs folder

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:accessFolderInfo/>
   <cx:digitalWork licensePartIdRef="Specs"/>
 </grant>
```

9.1.2 The Backup Right

Represents the right to create a backup copy of the work.

Schema Representation of the Backup Type

```
-<xsd:complexType name="Backup">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the backup Element

```
-<xsd:element name="backup" type="cx:Backup" substitutionGroup="r:right">
 </xsd:element>
```

Using the Backup Right

The Backup type represents the Right to make copies of a DigitalWork for the purpose of guarding against the loss of the original due to accident or catastrophic media or equipment failure

The backup copy is created as a new work, separate from the original work that was backed up. The only Right which may be exercised on a backup work is the Restore Right. The license for the original work may or may not be valid for the backup copy. For this reason, this specification does not mandate any specification of rights for the backup copy.

Backup is one of the file management rights.

Alice can Backup book1

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:backup/>
   <cx:digitalWork licensePartIdRef="book1"/>
 </grant>
```

9.1.3 The Copy Right

Represents the right to copy a work.

Schema Representation of the Copy Type

```
-<xsd:complexType name="Copy">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the copy Element

```
-<xsd:element name="copy" type="cx:Copy" substitutionGroup="r:right">
 </xsd:element>
```

Using the Copy Right

The new copy of the DigitalWork is created as a new work, separate from the original that was copied. The license for the original work may or may not be valid for the new copy. For this reason, this specification does not mandate any specification of rights for the new copy.

Copy is one of the transport rights.

Granting the Copy Right to distributors

```
-<grant>
  -<forAll varName="anyone">
   </forAll>
  -<keyHolder licensePartIdRef="issuedToParty">
   </keyHolder>
   <issue/>
  -<grant>
```

```
    <principal varRef="anyone"/>
    <cx:play/>
   -<digitalResource>
    -<dsig:Reference URI="http://www.server.com/downloads/anInterestingSong.mp3">
       <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <dsig:DigestValue>qZk+NkcGgWq6PiVxeFDCbJzQ2J0=</dsig:DigestValue>
      </dsig:Reference>
    </digitalResource>
  </grant>
 -<allConditions>
  -<sx:exerciseLimit>
   -<sx:stateReference>
    -<uddi>
     -<serviceKey>
        <uuid>D04951E4-332C-4693-B7DB-D3D1D1C20844</uuid>
      </serviceKey>
     </uddi>
    </sx:stateReference>
   </sx:exerciseLimit>
  -<validityInterval>
    <notBefore>2001-05-25T00:00:00</notBefore>
    <notAfter>2002-05-25T00:00:00</notAfter>
   </validityInterval>
  -<prerequisiteRight>
   -<keyHolder licensePartIdRef="issuedToParty">
    </keyHolder>
    <possessProperty/>
    <murphy:distributor/>
   -<trustedIssuer>
    -<keyHolder>
     -<info>
      -<dsig:KeyValue>
       -<dsig:RSAKeyValue>
          <dsig:Modulus>p8sN4KeeR...</dsig:Modulus>
          <dsig:Exponent>AQABAA==</dsig:Exponent>
        </dsig:RSAKeyValue>
       </dsig:KeyValue>
      </info>
     </keyHolder>
    </trustedIssuer>
   </prerequisiteRight>
  </allConditions>
 </grant>
```

9.1.4 The Delete Right

The right to delete a copy of a work from the repository.

Schema Representation of the `Delete` Type

```
 -<xsd:complexType name="Delete">
  -<xsd:complexContent>
    <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `delete` Element

```
 -<xsd:element name="delete" type="cx:Delete" substitutionGroup="r:right">
  </xsd:element>
```

Using the `Delete` `Right`

Generally, any copy owner would have the `Right` to `Delete` the `DigitalWork`.

The `Right` to `Delete` must be controlled if many people can log into a repository and `Delete` files either accidentally or in malicious mischief. To prevent the unwanted and unauthorized deletion of remotely-accessed `DigitalWork`s, a `Delete` `Right` typically includes various conditions.

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

`Delete` is one of the file management rights.

Alice is allowed to `Delete` the Specs folder

```
 -<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:delete/>
   <cx:digitalWork licensePartIdRef="Specs"/>
 </grant>
```

9.1.5 The Edit Right

Represents the right to make changes to a work to create a new work based on the original.

Schema Representation of the `Edit` Type

```
-<xsd:complexType name="Edit">
 -<xsd:complexContent>
    <xsd:extension base="r:Right"/>
  </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `edit` Element

```
-<xsd:element name="edit" type="cx:Edit" substitutionGroup="r:right">
 </xsd:element>
```

Using the `Edit` `Right`

Since the content of the edited work has changed from the original `DigitalWork`, the license for the original work may or may not be valid for the edited work. For this reason, when the edited copy is saved, it is considered a new work even if it overwrites the original copy. In addition, this specification does not mandate any specification of rights for the edited copy.

`Edit` is like `Extract` in that it creates a new work. It differs from `Extract` in that it confers the right to make changes to the work.

`Edit` is one of the derivative work rights.

Alice can `Edit` XMLBook

```
-<grant>
 -<keyHolder licensePartIdRef="Alice">
  </keyHolder>
  <cx:edit/>
  <cx:digitalWork licensePartIdRef="XMLbook"/>
 -<cx:helper>
  -<keyHolder>
   -<info>
    -<dsig:KeyValue>
     -<dsig:RSAKeyValue>
        <dsig:Modulus>g8NRYMG30...</dsig:Modulus>
        <dsig:Exponent>AQABAA==</dsig:Exponent>
      </dsig:RSAKeyValue>
    </dsig:KeyValue>
   </info>
  </keyHolder>
 </cx:helper>
</grant>
```

9.1.6 The Embed Right

Represents the right to include the work as part of a composite work. An Embed operation places a copy of the work inside the composite work.

Schema Representation of the `Embed` Type

```
-<xsd:complexType name="Embed">
 -<xsd:complexContent>
    <xsd:extension base="r:Right"/>
  </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `embed` Element

```
-<xsd:element name="embed" type="cx:Embed" substitutionGroup="r:right">
 </xsd:element>
```

Using the `Embed` `Right`

The `Embed` type represents the `Right` to include a `DigitalWork` as part of another work, forming a composite work.

The composite work is created as a new work, separate from the original work that was embedded. The license for the original work may or may not be valid for the composite work. For this reason, this specification does not mandate any specification of rights for the composite work.

`Embed` is one of the derivative work rights.

XMLBook may be embedded in another work

```
-<grant>
```

```
 -<keyHolder licensePartIdRef="Alice">
  </keyHolder>
  <cx:embed/>
  <cx:digitalWork licensePartIdRef="XMLbook"/>
 </grant>
```

9.1.7 The Execute Right

Represents the right to execute a resource from a secure repository.

Schema Representation of the `Execute` Type

```
-<xsd:complexType name="Execute">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `execute` Element

```
-<xsd:element name="execute" type="cx:Execute" substitutionGroup="r:right">
 </xsd:element>
```

Using the `Execute` `Right`

`Execute` is one of the file management rights.

Alice can `execute` the XmlReader application

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:execute/>
   <cx:digitalWork licensePartIdRef="XMLReader"/>
 </grant>
```

9.1.8 The Export Right

Represents the right to make a source copy of the work outside of the secure repository.

Schema Representation of the `Export` Type

```
-<xsd:complexType name="Export">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `export` Element

```
-<xsd:element name="export" type="cx:Export" substitutionGroup="r:right">
 </xsd:element>
```

Using the `Export` `Right`

Typically, the exported copy of the `DigitalWork` would be in a file format suitable for unrestricted viewing, printing, or editing. Thus, this `Right` can be used to make a digital copy that is not encrypted or otherwise protected. For example, an `Export` `Right` might be exercised to release an older work after it has passed out of copyright.

The exported copy is created as a new work, separate from the original work. The license for the original work may or may not be valid for the exported copy. For this reason, this specification does not mandate any specification of rights for the exported copy.

`Export` differs from `Copy` in that an `Export` is made to an in-the-clear non-secure repository whereas a `Copy` is made to another secure repository.

`Export` is one of the render rights.

Alice can `Export` chapter 1 if she pays $30.00

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:export/>
   <cx:digitalWork licensePartIdRef="chapter1"/>
  -<sx:fee>
   -<sx:paymentPerUse>
      <sx:rate>30.00</sx:rate>
    </sx:paymentPerUse>
```

```
    -<sx:to>
      -<sx:aba>
         <sx:institution>12345</sx:institution>
         <sx:account>98765</sx:account>
       </sx:aba>
     </sx:to>
   </sx:fee>
 </grant>
```

9.1.9 The Extract Right

Represents the right to use a portion of the work to create a new work.

Schema Representation of the `Extract` Type

```
-<xsd:complexType name="Extract">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `extract` Element

```
-<xsd:element name="extract" type="cx:Extract" substitutionGroup="r:right">
 </xsd:element>
```

Using the `Extract` `Right`

A rights owner can divide a `DigitalWork` up into several sub-works, each with its own rights specification. In this way, the rights owner can decide whether a work can be reused as a whole or in parts and associate different `Right`s and `Condition`s with the parts of a `DigitalWork`.

The extracted material is created as a new work, separate from the original work. The license for the original work may or may not be valid for the extracted copy. For this reason, this specification does not mandate any specification of rights for the extracted copy.

`Extract` differs from `Edit` in that it does not grant the `Right` to modify a work.

`Extract` is one of the derivative work rights.

Alice can `Extract` image1

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:extract/>
   <cx:digitalWork licensePartIdRef="image1"/>
 </grant>
```

9.1.10 The Install Right

Represents the right to make software runable on the repository, including, for example, checking that the software is certified, that it has not been tampered with, and that it is compatible with the repository.

Schema Representation of the `Install` Type

```
-<xsd:complexType name="Install">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `install` Element

```
-<xsd:element name="install" type="cx:Install" substitutionGroup="r:right">
 </xsd:element>
```

Using the `Install` `Right`

Simply copying a program to a repository does not make it runable. 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.

`Install` is one of the configuration rights.

Alice is granted the `Right` to `Install` a Setup program

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
```

```
    </keyHolder>
    <cx:install/>
    <cx:digitalWork licensePartIdRef="Setup"/>
  </grant>
```

9.1.11 The Loan Right

Represents the right to lend a work to another principal for a specific period of time. While the work is on loan, the original copy cannot be used.

Schema Representation of the `Loan` Type

```
-<xsd:complexType name="Loan">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `loan` Element

```
-<xsd:element name="loan" type="cx:Loan" substitutionGroup="r:right">
 </xsd:element>
```

Using the `Loan` `Right`

Exercising a `Loan` `Right` creates a "loaner" copy of a `DigitalWork` on a receiving repository. The loaner copy is created as new work, separate from the original work. The `license` for the original work may or may not be valid for the loaner copy. For this reason, this specification does not mandate any specification of rights for the loaner copy.

Typically, the original copy of the work cannot be used while the work is "on loan". Throughout the loan period, both repositories must take the loan into account in all transactions relevant to the work on loan. At the end of the loan period, the loaner copy deactivates and the original copy reactivates.

Note: If the original repository contains more than one copy of the work, the original repository can still exercise all rights on copies that are not on loan.

`Loan` is one of the transport rights.

The loaner copy may be played by anyone for 1 month

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:loan/>
   <cx:digitalWork licensePartIdRef="book1"/>
 </grant>
```

9.1.12 The ManageFolder Right

Schema Representation of the `ManageFolder` Type

```
-<xsd:complexType name="ManageFolder">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `manageFolder` Element

```
-<xsd:element name="manageFolder" type="cx:ManageFolder" substitutionGroup="r:right">
 </xsd:element>
```

Using the `ManageFolder` `Right`

The `ManageFolder` type represents the `Right` to perform the following operations:

- create subfolders
- name subfolders
- reconfigure folders by moving files and folders among folders

The `Right` to perform all of these repository actions is governed by the single `ManageFolder` `Right`; there are no separate rights for moving or renaming files and folders. The `ManageFolder` `Right` is commonly exercised by commands at a repository user interface.

`ManageFolder` is one of the file management rights.

Granting `ManageFolder` access

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:manageFolder/>
   <cx:digitalWork licensePartIdRef="Specs"/>
 </grant>
```

9.1.13 The Play Right

Represents the right to render the work in a transient form, as appropriate to the content type. Depending on the content type, exercising this right might result in displaying a book, playing an audio clip, or showing a video.

Schema Representation of the `Play` Type

```
-<xsd:complexType name="Play">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `play` Element

```
-<xsd:element name="play" type="cx:Play" substitutionGroup="r:right">
 </xsd:element>
```

Using the `Play` `Right`

Typically, a `Grant` containing a `Play` element also contains a `Condition` specifying the device on which to `Play` the `Resource`. For more information, refer to the `Renderer` `Condition`.

`Play` is one of the render rights.

Grant someone the right to `Play` a movie

```
-<grant>
  -<forAll varName="anyone">
   </forAll>
   <principal varRef="anyone"/>
   <cx:play/>
  -<cx:digitalWork>
    -<cx:metadata>
      -<xml>
        -<cx:simpleDigitalWorkMetadata>
           <cx:title>Air Force One</cx:title>
         </cx:simpleDigitalWorkMetadata>
       </xml>
     </cx:metadata>
    -<cx:locator>
      -<secureIndirect URI="http://sonyPictures.com/AirForceOne">
         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <dsig:DigestValue>0kccZ4a3zFW9OPT1qEIqSg==</dsig:DigestValue>
       </secureIndirect>
     </cx:locator>
   </cx:digitalWork>
  -<sx:fee>
    -<sx:paymentPerUse>
       <sx:rate currency="USD">2.00</sx:rate>
     </sx:paymentPerUse>
    -<sx:to>
      -<sx:aba>
         <sx:institution>139371581</sx:institution>
         <sx:account>111111</sx:account>
       </sx:aba>
     </sx:to>
   </sx:fee>
 </grant>
```

9.1.14 The Print Right

Represents the right to make a permanent non-digital rendered copy of the work outside the control of a repository. Exercising this right might result in printing a hard copy of a book or creating an audio recording on a magnetic tape.

Schema Representation of the `Print` Type

```
-<xsd:complexType name="Print">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `print` Element

```
-<xsd:element name="print" type="cx:Print" substitutionGroup="r:right">
 </xsd:element>
```

Using the `Print` `Right`

`Print` is different from `Export` in that `Print` creates non-digital copies while `Export` creates digital copies.

`Print` is one of the render rights.

Grant someone the `Right` to `Print` a book

```
-<grant>
 -<keyHolder>
  -<info>
   -<dsig:KeyValue>
    -<dsig:RSAKeyValue>
       <dsig:Modulus> p8sN4Kee...</dsig:Modulus>
       <dsig:Exponent>AQABAA==</dsig:Exponent>
     </dsig:RSAKeyValue>
    </dsig:KeyValue>
  </info>
 </keyHolder>
 <cx:print/>
 -<cx:digitalWork>
   <cx:description>Alice In the Wonder Land</cx:description>
 </cx:digitalWork>
 -<allConditions>
  -<school:content>
    <school:unit type="onix:NumberOfPages"/>
    <school:from>1</school:from>
    <school:to>10</school:to>
   </school:content>
  -<cx:watermark>
    <cx:user-name/>
    <cx:string>Title: 'Alice In the Wonder Land'</cx:string>
    <cx:render-location/>
    <cx:render-time/>
    <cx:object licensePartIdRef="logo"/>
   </cx:watermark>
 </allConditions>
</grant>
```

9.1.15 The Read Right

Represents the right to read the work from the repository.

Schema Representation of the `Read` Type

```
-<xsd:complexType name="Read">
 -<xsd:complexContent>
    <xsd:extension base="r:Right"/>
  </xsd:complexContent>
</xsd:complexType>
```

Schema Representation of the `read` Element

```
-<xsd:element name="read" type="cx:Read" substitutionGroup="r:right">
 </xsd:element>
```

Using the `Read` `Right`

The `Read` type represents a `Right` to *access* a `DigitalWork`. It should not be confused with `Play`, which represents the `Right` to *render* a `DigitalWork`.

`Read` is one of the file management rights.

Granting the `Right` to `Read`

```
-<grant>
 -<keyHolder licensePartIdRef="Alice">
  </keyHolder>
  <cx:read/>
  <cx:digitalWork licensePartIdRef="Specs"/>
</grant>
```

9.1.16 The Restore Right

Represents the right to restore a work from a backup copy, converting the backup copy to usable form.

Schema Representation of the `Restore` Type

```
-<xsd:complexType name="Restore">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `restore` Element

```
-<xsd:element name="restore" type="cx:Restore" substitutionGroup="r:right">
 </xsd:element>
```

Jsing the `Restore` Right

Restore is one of the file management rights.

Granting the `Right` to `Restore` a backup copy

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:restore/>
   <cx:digitalWork licensePartIdRef="book1"/>
 </grant>
```

9.1.17 The Transfer Right

Represents the right to transfer a work to another repository, removing the work from the original location.

Schema Representation of the `Transfer` Type

```
-<xsd:complexType name="Transfer">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `transfer` Element

```
-<xsd:element name="transfer" type="cx:Transfer" substitutionGroup="r:right">
 </xsd:element>
```

Jsing the `Transfer` `Right`

Exercising a `Transfer` `Right` moves the `DigitalWork` from one repository to another. Exercising a `Transfer` `Right` does not ncrease the number of copies of a work, because the `Transfer` transaction between two repositories removes the `DigitalWork` rom the original repository when the copy has been created and verified on the receiving repository.

The transferred copy is created as a new work in the receiving repository, separate from the original work. The license for the original work may or may not be valid for the transferred copy. For this reason, this specification does not mandate any specification of rights or the transferred copy.

Transfer is one of the transport rights.

Alice can `Transfer` book1

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:transfer/>
   <cx:digitalWork licensePartIdRef="book1"/>
 </grant>
```

9.1.18 The Uninstall Right

Represents the right to disable software from running, restoring it to the state it was in before it was installed. Exercising an uninstall ight does not remove the files for the program from the repository.

Schema Representation of the `Uninstall` Type

```
-<xsd:complexType name="Uninstall">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `Uninstall` Element

```
-<xsd:element name="uninstall" type="cx:Uninstall" substitutionGroup="r:right">
```

```
  </xsd:element>
```

Using the Uninstall Right

The Uninstall type represents the Right to Uninstall a Resource. The Uninstall operation removes software from the running system. The Uninstall operation does not Delete the file corresponding to the program; it merely disables the program from running, restoring it to the state in which it was before installation.

Uninstall is one of the configuration rights.

Granting Alice the Right to Uninstall XMLReader

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
    </keyHolder>
    <cx:uninstall/>
    <cx:digitalWork licensePartIdRef="XMLReader"/>
  </grant>
```

9.1.19 The Verify Right

Represents the right to check the authenticity of the work in the repository.

Schema Representation of the Verify Type

```
-<xsd:complexType name="Verify">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the Verify Element

```
-<xsd:element name="verify" type="cx:Verify" substitutionGroup="r:right">
 </xsd:element>
```

Using the Verify Right

The Verify type represents the Right to authenticate a given DigitalWork and/or verify its integrity.

- Authentication ensures that a work comes from the intended source. Authentication typically involves verifying signatures of signed data using the public/private key pair.
- Integrity verification ensures that the data received exactly matches the data that was sent; it ensures that the data has not been tampered with. Integrity verification involves using one-way hash functions and other tamper detection keys.

Verify is one of the file management rights.

Granting the Verify right

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
    </keyHolder>
    <cx:verify/>
    <cx:digitalWork licensePartIdRef="book1"/>
  </grant>
```

9.1.20 The Write Right

Represents the right to write or save the work in the secure repository.

Schema Representation of the Write Type

```
-<xsd:complexType name="Write">
  -<xsd:complexContent>
     <xsd:extension base="r:Right"/>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the Write Element

```
-<xsd:element name="write" type="cx:Write" substitutionGroup="r:right">
 </xsd:element>
```

Using the Write Right

Write is one of the file management rights.

Granting the Write right

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:write/>
   <cx:digitalWork licensePartIdRef="Specs"/>
 </grant>
```

9.2 Content Extension Resources and Metadata

This section describes each of the Resource and metadata elements defined by the XrML Content Extension. These elements represent DigitalWorks and the information about them.

9.2.1 The DigitalWork Resource

A resource that represents the content to which rights and conditions are being applied.

Schema Representation of the DigitalWork Type

```
-<xsd:complexType name="DigitalWork">
  -<xsd:complexContent>
    -<xsd:extension base="r:Resource">
      -<xsd:sequence minOccurs="0">
        -<xsd:element name="description" type="r:LinguisticString" minOccurs="0" maxOccurs="unbounded">
         </xsd:element>
        -<xsd:element name="metadata" type="r:DigitalResource" minOccurs="0" maxOccurs="unbounded">
         </xsd:element>
        -<xsd:element name="locator" type="r:DigitalResource" minOccurs="0">
         </xsd:element>
        -<xsd:element name="parts" minOccurs="0">
          -<xsd:complexType>
            -<xsd:sequence>
               <xsd:element ref="cx:digitalWork" minOccurs="0" maxOccurs="unbounded"/>
             </xsd:sequence>
           </xsd:complexType>
         </xsd:element>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the digitalWork Element

```
-<xsd:element name="digitalWork" type="cx:DigitalWork" substitutionGroup="r:resource">
 </xsd:element>
```

Using the DigitalWork Resource

A DigitalWork may consist of several identified parts, each of which is itself a DigitalWork. With this construction, Rights to use different parts of the work might be granted differently to different parties.

Specifying a book as a DigitalWork

```
-<cx:digitalWork>
  -<cx:metadata>
    -<xml>
      -<cx:simpleDigitalWorkMetadata>
         <cx:title>Alice in the Wonder Land</cx:title>
         <cx:copyright>Copyright 1999 Addison Wesley. All Rights Reserved.</cx:copyright>
         <onix:mediaFileTypeCode>movie0001</onix:mediaFileTypeCode>
       </cx:simpleDigitalWorkMetadata>
     </xml>
   </cx:metadata>
  -<cx:locator>
    -<secureIndirect URI="http://sonyPictures.com/AliceInTheWonderLand">
       <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
       <dsig:DigestValue>0kccZ4a3zFW9OPT1qEIqSg==</dsig:DigestValue>
     </secureIndirect>
   </cx:locator>
 </cx:digitalWork>
```

9.2.1.1 description

The optional description element describes the work, possibly in different languages. The sequence of bits identified by this DigitalWork is not affected by this element.

9.2.1.2 metadata

The optional metadata element enables referencing of metadata that is embedded in a Resource, or metadata that exists in a database, or an XML fragment of metadata that makes use of other metadata schemes. It is of type DigitalResource defined in

the core schema. The sequence of bits identified by this DigitalWork must be able to be shown to have each of the metadata specified here.

9.2.1.4 locator

The optional locator element specifies how to locate the content of the work. It may specify a location or an inline inclusion of the content. It is of type DigitalResource, which is defined in the XrML Core. The sequence of bits identified by this DigitalWork must be that sequence of bits identified by this locator, whose semantics are exactly that of its type.

9.2.1.5 parts

The optional parts element specifies the DigitalWorks that are included as parts of this work. It is of type DigitalWork. The sequence of bits identified by this DigitalWork must be able to be shown to have each of the parts specified here.

Specifying Chapter 1 as part of a book

```
-<cx:parts>
 -<cx:digitalWork licensePartId="chapter1">
  -<cx:metadata>
   -<xml>
    -<cx:simpleDigitalWorkMetadata>
       <cx:title>Chapter 1: In the Beginning</cx:title>
     </cx:simpleDigitalWorkMetadata>
   </xml>
  </cx:metadata>
 </cx:digitalWork>
</cx:parts>
```

9.2.2 The SimpleDigitalWorkMetadata

Schema Representation of the SimpleDigitalWorkMetadata Type

```
-<xsd:complexType name="SimpleDigitalWorkMetadata">
 -<xsd:sequence>
  -<xsd:element name="title" type="r:LinguisticString" minOccurs="0">
   </xsd:element>
  -<xsd:element name="creator" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
   </xsd:element>
  -<xsd:element name="publisher" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
   </xsd:element>
  -<xsd:element name="publicationDate" type="xsd:dateTime" minOccurs="0">
   </xsd:element>
  -<xsd:element name="owner" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
   </xsd:element>
  -<xsd:element name="copyright" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
   </xsd:element>
  -<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded">
   </xsd:any>
 </xsd:sequence>
</xsd:complexType>
```

Schema Representation of the simpleDigitalWorkMetadata Element

```
-<xsd:element name="simpleDigitalWorkMetadata" type="cx:SimpleDigitalWorkMetadata">
 </xsd:element>
```

Using the SimpleDigitalWorkMetadata

The SimpleDigitalWorkMetadata type may be embedded in the metadata xml specification to specify information related to the DigitalResource. Note that the xml element is part of the DigitalResource type, which servers the type of metadata. DigitalResource is defined in XrML Core.

9.2.2.1 title

The SimpleDigitalWorkMetadata title specifies the work title; it has type LinguisticString, which is defined in the XrML Core.

9.2.2.2 creator

The creator element specifies the party who created the work, such as an author, illustrator, editor or translator.

9.2.2.3 publisher

The publisher element specifies the party who published the work.

9.2.2.4 publicationDate

The publicationDate specifies the date (and possibly time) at which this work was published.

9.2.2.5 owner

The `owner` element specifies the holder of the copyright for the work.

9.2.2.6 copyright

The `copyright` element specifies the formal copyright declaration(s) of the work.

Specifying simple metadata for a book. Note the use of the `MediaFileTypeCode` entity from ONIX.

```
-<xml>
 -<cx:simpleDigitalWorkMetadata>
    <cx:title>Alice in the Wonder Land</cx:title>
    <cx:copyright>Copyright 1999 Addison Wesley. All Rights Reserved.</cx:copyright>
    <onix:mediaFileTypeCode>movie0001</onix:mediaFileTypeCode>
  </cx:simpleDigitalWorkMetadata>
 </xml>
```

9.2.3 The SecurityLevel Resource

An abstract indication of the security level that a principal has. Specified as the resource in a grant that also specifies the possessProperty right.

Schema Representation of the `SecurityLevel` Type

```
-<xsd:complexType name="SecurityLevel" abstract="true">
 -<xsd:complexContent>
   -<xsd:extension base="r:Resource">
     -<xsd:sequence>
       -<xsd:element name="value" type="xsd:integer">
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `securityLevel` Element

```
-<xsd:element name="securityLevel" type="cx:SecurityLevel" substitutionGroup="r:resource">
 </xsd:element>
```

Using the `SecurityLevel` `Resource`

To specify a `SecurityLevel`, place the `securityLevel` element as a `Resource` in a `Grant` that also specifies the `PossessProperty` `Right`, which is defined in the XrML Core.

The `SecurityLevel` element is an extension of the `Resource` element, which is defined in the XrML Core.

9.2.3.1 value

The `value` element specifies the security level.

Assigning a `securityLevel` of 5 to a projector

```
-<grant>
 -<forAll varName="projectorEpson">
   -<everyone>
     -<school:projectorProperties>
        <school:resolution>SXGA</school:resolution>
        <school:manufacturer>EPSON</school:manufacturer>
        <school:sizeCategory>Portable</school:sizeCategory>
      </school:projectorProperties>
     -<trustedIssuer>
       -<keyHolder licensePartIdRef="trustedIssuer">
        </keyHolder>
      </trustedIssuer>
    </everyone>
  </forAll>
 -<keyHolder varRef="projectorEpson">
  </keyHolder>
  <possessProperty/>
 -<school:kernelSecurityLevel>
    <cx:value>5</cx:value>
  </school:kernelSecurityLevel>
 </grant>
```

9.3 Content Extension Conditions and Obligations

This section describes each of the condition and obligation elements defined by the XrML Content Extension. These elements represent restrictions or modifiers that can be associated with rights in `Grant` specifications.

9.3.1 The Destination Condition

Indicates the repositories to which a work can be moved. Used with rights that involve movement of digital works (all rights except render rights).

Schema Representation of the Destination Type

```
-<xsd:complexType name="Destination">
  -<xsd:complexContent>
    -<xsd:extension base="r:Condition">
      -<xsd:sequence>
         <xsd:element ref="r:principal"/>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the destination Element

```
-<xsd:element name="destination" type="cx:Destination" substitutionGroup="r:condition">
 </xsd:element>
```

Using the Destination Condition

A typical use of a Destination element is in a Grant that includes a Transfer Right. In this case, the Destination element could be used to limit the secure repository to which a work can be transferred.

The Destination type is an extension of the Condition type, which is defined in the XrML Core.

The Destination/principal must be the destination secure repository for this Condition to be satisfied.

Specifying the Destination of a ManageFolder Right

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:manageFolder/>
   <cx:digitalWork licensePartIdRef="Specs"/>
  -<allConditions>
    -<cx:source>
      -<keyHolder licensePartIdRef="SmartCard">
       </keyHolder>
     </cx:source>
    -<cx:destination>
      -<keyHolder licensePartIdRef="SmartCard">
       </keyHolder>
     </cx:destination>
   </allConditions>
 </grant>
```

9.3.2 The Helper Condition

Indicates the software that can be used to exercise a right.

Schema Representation of the Helper Type

```
-<xsd:complexType name="Helper">
  -<xsd:complexContent>
    -<xsd:extension base="r:Condition">
      -<xsd:sequence>
         <xsd:element ref="r:principal"/>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the helper Element

```
-<xsd:element name="helper" type="cx:Helper" substitutionGroup="r:condition">
 </xsd:element>
```

Using the Helper Condition

The Helper type may be specified for any Right. A typical use of a Helper element is in a Grant that includes a Play right. In this case, the Helper element could be used to specify the software that can be used to Play the work.

The Helper type is an extension of the Condition type, which is defined in the XrML core.

The Helper/principal must be the controlling software for this Condition to be satisfied.

Specifying the software to be used for editing

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:edit/>
   <cx:digitalWork licensePartIdRef="XMLbook"/>
  -<cx:helper>
    -<keyHolder>
      -<info>
        -<dsig:KeyValue>
          -<dsig:RSAKeyValue>
             <dsig:Modulus>g8NRYMG30...</dsig:Modulus>
             <dsig:Exponent>AQABAA==</dsig:Exponent>
           </dsig:RSAKeyValue>
         </dsig:KeyValue>
       </info>
     </keyHolder>
   </cx:helper>
 </grant>
```

9.3.3 The Renderer Condition

Identifies the device that can be used to render a work. Used with render rights.

Schema Representation of the `Renderer` Type

```
-<xsd:complexType name="Renderer">
  -<xsd:complexContent>
    -<xsd:extension base="r:Condition">
      -<xsd:sequence>
         <xsd:element ref="r:principal"/>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `renderer` Element

```
-<xsd:element name="renderer" type="cx:Renderer" substitutionGroup="r:condition">
 </xsd:element>
```

Using the `Renderer` `Condition`

A typical use of a `Renderer` element is in a `Grant` that includes an `Play` `Right`. In this case, the `Renderer` element could be used to specify the device on which the work can be played.

The `Renderer` type is an extension of the `Condition` type, which is defined in the XrML Core.

The `Renderer`/`principal` must be the rendering device for this `Condition` to be satisfied.

Specifying an authorized projector as a `Renderer`

```
-<cx:renderer>
  -<keyHolder varRef="projectorEpson">
   </keyHolder>
 </cx:renderer>
```

9.3.4 The Source Condition

Indicates the source secured repository or device to use when exercising a right. Used with all rights except for render rights.

Schema Representation of the `Source` Type

```
-<xsd:complexType name="Source">
  -<xsd:complexContent>
    -<xsd:extension base="r:Condition">
      -<xsd:sequence>
         <xsd:element ref="r:principal"/>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the `source` Element

```
-<xsd:element name="source" type="cx:Source" substitutionGroup="r:condition">
 </xsd:element>
```

Using the `Source` `Condition`

A typical use of a Source element is in a Grant that includes a Copy Right. In this case, the Source element could be used to limit the source device from which to read.

The Source type is an extension of the Condition type, which is defined in the XrML Core.

The Source/principal must be the source secure repository for this Condition to be satisfied.

Alice is granted ManageFolder right on a smartcard

```
-<grant>
  -<keyHolder licensePartIdRef="Alice">
   </keyHolder>
   <cx:manageFolder/>
   <cx:digitalWork licensePartIdRef="Specs"/>
  -<allConditions>
    -<cx:source>
      -<keyHolder licensePartIdRef="SmartCard">
       </keyHolder>
     </cx:source>
    -<cx:destination>
      -<keyHolder licensePartIdRef="SmartCard">
       </keyHolder>
     </cx:destination>
   </allConditions>
 </grant>
```

9.3.5 The Watermark Condition

A list of information to be embedded in a copy of the work by a device while producing this copy.

Schema Representation of the Watermark Type

```
-<xsd:complexType name="Watermark">
  -<xsd:complexContent>
    -<xsd:extension base="r:Condition">
      -<xsd:sequence minOccurs="0">
        -<xsd:choice minOccurs="0" maxOccurs="unbounded">
          -<xsd:element name="string" type="r:LinguisticString">
           </xsd:element>
           <xsd:group ref="cx:WatermarkToken"/>
          -<xsd:element name="object" type="cx:DigitalWork">
           </xsd:element>
         </xsd:choice>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
 </xsd:complexType>
```

Schema Representation of the watermark Element

```
-<xsd:element name="watermark" type="cx:Watermark" substitutionGroup="r:condition">
 </xsd:element>
```

Using the Watermark Condition

The Watermark Condition is typically used in Grants that specify Backup, Export, Install, Play, and Print rights.

9.3.5.1 string

The string element specifies a piece of information known at the time the DigitalWork is published, such as distributor contact URL or creator ID.

9.3.5.2 WatermarkToken

The WatermarkToken element represents a fingerprint watermark. The fingerprint watermark ensures copyright protection by watermarking the DigitalWork with customer identification used to track and trace legal or illegal copies. The following table lists the defined watermark tokens and their meanings:

Token	Meaning
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

Watermark in printed copy

```
-<cx:watermark>
   <cx:user-name/>
   <cx:string>Title: 'Alice In the Wonder Land'</cx:string>
   <cx:render-location/>
   <cx:render-time/>
   <cx:object licensePartIdRef="logo"/>
 </cx:watermark>
```

9.3.5.3 Object

The `object` element specifies a kind of watermark that is a `DigitalWork` encoded so that it cannot be found without possession of a secret key.

Go to Part V: Appendices