Adobe

Portable Document Rights Language (PDRL) Reference

Adobe® LiveCycle® ES3

April 11, 2012Version 10.0.2

Portable Document Rights Language (PDRL) Reference

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Contents

About this help

This Help explains Portable Document Rights Language (PDRL), which is a language for expressing rights and conditions for accessing digital content. PDRL meets the following business requirements:

Document control: Allows for the expression of the document control features implemented in Adobe® LiveCycle® Rights Management 10.

Security: Can be used for securely transmitting and attaching rights to protected resources.

Extensibility: Can be extended to handle new document control features.

Compatibility with PDF: Compatible with the PDF public specification [PDF]. Policies and licenses described within this document can be contained within a PDF file.

Additional information

The information in this table will help you to learn more about Adobe LiveCycle ES3.

For information about	See
LiveCycle ES3 solution components	LiveCycle ES3 Overview
The features available with each service	LiveCycle ES3 Services

1 PDRL Reference

Introducing PDRL

PDRL incorporates the following design principles:

Abstract: PDRL provides the building blocks for communicating rights and conditions for access to protected resources. Although the first target implementation for this language is document control through Rights Management 10, it is abstract enough for use in expressing other types of policies in other implementations. For example, a *resource* is a very generic concept. Rights Management 10 defines a *resource* to be a document. However, a *resource* could just as easily be video, music, or some other form of digital content.

Expressive: PDRL provides enough richness in the language to support a number of document control features. Therefore, the language is designed in an object-oriented manner, with a key set of building block objects like conditions, permissions, policy entries, principals, and properties to allow for creation of sophisticated policies.

Extensible: PDRL provides several mechanisms for extensibility. The goal was to define the basic containers and building blocks in the core schema. Then, as new features are needed, they can be added in separate schema extension documents without need for modification to the core language.

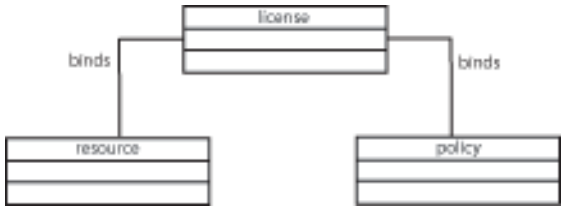
Interoperable: The rights management space is still relatively new. There is no clear industry standard for rights language at this point. For this reason, each vendor has defined its own notion of policies. For Rights Management 10 to be successful, we must be able to convert between other rights languages and PDRL without loss of data. PDRL was designed with this goal in mind.

XML Schema: PDRL was defined using XML Schema. XML and XML Schema were chosen as the syntax to express this language because of their pervasiveness and wide acceptance.

Non-implementation-specific: PDRL is as implementation-neutral as possible so that it can be used in future implementations without change. It also makes the interoperability goal more achievable.

2 PDRL Concepts

PDRL consists of three primary XML elements: *license*, *policy*, and *resource*. These primary objects are combined to define the rights and policies for accessing a resource. All of the other elements and attributes defined within this specification are sub-elements to these three.



Example use case

Adobe LiveCycle Rights Management 10 and its integration with Adobe Acrobat® 8.0 is provided as an example of how these elements work together.

Note: PDRL can also be used by other policy servers to describe rights for non-PDF documents.

A user logs into Rights Management 10 and creates a policy that has the following properties:

- A list of users and groups (also known as *principals*) that can access a document; the permissions for each user or group.
- The expiration time on the document, if any.
- Other miscellaneous document control settings, such as auditing.

The *policy* is a stand-alone object. It is converted to XML (PDRL) and stored in the Rights Management 10 database.

A user can then open a PDF document using Acrobat and apply security to that document. The Acrobat user interface displays a list of existing policies. The user selects one of these policies based on the security needed for the business function and applies the policy to the document.

At this point, Acrobat contacts Rights Management 10. Rights Management 10 creates two new objects to complete the processing. The first object is a *resource* object. It contains metadata to identify the document being secured and the publisher of the document. The second object is a *license* object. As shown in the preceding figure, the *license* object is the binding between a *resource* object and a *policy* object. It represents the fact that a specific document has been secured and is now controlled based on the rules defined in the policy. All of these objects can exist independently from one another and can be converted to or from XML using the PDRL schema. The license is the glue that binds a resource to a policy and describes the exact rules used to control access to that document instance. Note that multiple resources can be controlled by the same policy. Also, a policy object's rules can be changed dynamically, and all resources controlled by that policy will automatically enforce the new rules due to this indirection in the model.

License

A *license* contains the information to bind a secured resource to a policy. It gets created at the time of securing a document and consists of the *issuing authority*, *resource*, *policy* or *policy ID*, and *signature*. It specifies the rights available to a principal who attempts to access a secured resource. A license should be generated by a security system as part of the process of applying controls to a resource. The security system (or *issuing authority*) issues the license to encapsulate the fact that some kind of cryptographic controls have been put in place around that particular resource. The license can then be used in subsequent resource access operations by the security enforcement system to enforce the rights or policies that apply to the principal accessing the resource.

The license *must* include some kind of integrity protection, either a digital signature or an HMAC, to ensure that the binding has not been tampered with.

The following XML shows a license with a *detached* policy referenced by a policy ID. Policies may optionally be included within the license body.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<License
   LicenseSchemaVersion="1.0"
   LicenseID="21F70094-447F-07C5-EC13-01A6BEC4C2CC"
   LicenseInstanceVersion="1"
   xmlns="http://www.adobe.com/schema/1.0/pdrl">
   <IssuingAuthority>http://aps.corp.adobe.com:8080/axis/services/
      urn:EDCLicenseService</IssuingAuthority>
   <Resource>
      <Publisher PrincipalNameType="USER">
         <PrincipalDomain>adobe.com</PrincipalDomain>
         <PrincipalName>uid=
bshapiro,ou=people,o=adobe.com</PrincipalName>
      </Publisher>
      <PublishTime>2004-06-23T18:37:06.465-07:00</PublishTime>
      <ResourceName>importantspec.pdf</ResourceName>
      <ResourceID>975AFE5F-D9B3-E623-2A79-CCCFFFA2087E</ResourceID>
   </Resource>
   <PolicyIDReference
PolicyID="4F3F323D-5C45-3031-8A52-F6151AB82927"/>
   <HMAC>kpRyejY4uxwT9I74FYv8nQ==</HMAC>
</License>
```

This example shows that a license for resource "importantspec.pdf" was issued by the issuing authority "http://aps.corp.adobe.com:8080/axis/services/urn:EDCLicenseService" on 6/23/2004. It binds the resource to the policy with ID="4F3F323D-5C45-3031-8A52-F6151AB82927". The publisher of the resource was *bshapiro*.

Resource

A *resource* is a reference to the resource or object for which rights are being granted. In Rights Management 10, a *resource* is the document being controlled, but in other contexts it could refer to other content types.

The *resource* is a relatively simple data type. It contains an optional resource name and some form of resource identifier. This identifier can be either a URI or some sort of alphanumeric ID. The interpretation of this identifier is implementation-specific.

The following XML shows a resource definition.

```
<Resource>
   <Publisher PrincipalNameType="USER">
      <PrincipalDomain>ldap://corp.adobe.com</PrincipalDomain>
      <PrincipalName>bshapiro</PrincipalName>
   </Publisher>
   <PublishTime>2001-12-17T09:30:47-05:00</PublishTime>
   <ResourceName>importantspec.pdf</ResourceName>
   <ResourceID>975AFE5F-D9B3-E623-2A79-CCCFFFA2087E </ResourceID>
</Resource>
```

Policy

A *policy* is the container for the rules that determine the list of principals that can perform privileged operations on a secured resource. This is one of the primary elements in the model. A *policy* is defined independently of a resource being protected. Because policy definition is complex, the idea is to create a set of policies that can be reused when protecting new resources. A policy is not used to enforce rights until it is bound to a resource through the license.

At the most fundamental level, a policy lists the set of permissions or "rights" that principals have to a resource. Policies typically bundle together a set of rights that would be used to satisfy a single business requirement. For example, an administrator may create a policy named "Company Confidential" that ensures that corporate documents are only accessible to company employees. This policy would be applied to all documents that met those business criteria.

Another important point to note about policies is that they contain implementation-specific rights. The policy structure itself was designed to be very extensible. It can be used for defining policies for any arbitrary application. Therefore, the interpretation of a given permission or condition is really outside the scope of this specification. Instead, the language provides a basic set of policy building blocks within the core schema and then allows the extension of the schema to add implementation-specific rights.

A policy contains *principals*, *policy entries*, *conditions*, and *properties*. *Principals* are the actors on a resource. They are the users and groups of users trying to access controlled resources. *Policy entries* provide a way to build up a policy one piece at a time. They allow the flexibility

to give some principals different rights from others. *Conditions* are used to set restrictions on how and when a policy applies. One example of a condition is the date when access to a resource should expire. *Properties* provide an extensibility mechanism for adding name/value pairs that can potentially be used in the policy evaluation.

The following XML shows a typical policy.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Policy PolicyID="4F3F323D-5C45-3031-8A52-F6151AB82927"
PolicyInstanceVersion="1"
xmlns="http://www.adobe.com/schema/1.0/pdrl"
xmlns:pdrl-ex="http://www.adobe.com/schema/1.0/pdrl-ex"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.adobe.com/schema/1.0/pdrl-ex
..\pdrl-ex.xsd">
   <PolicyEntry>
      <Permission PermissionName="pdrl-ex:com.adobe.aps.onlineOpen"
Access="ALLOW"/>
      <Permission
PermissionName="pdrl-ex:com.adobe.aps.offlineOpen"
Access="ALLOW"/>
   <Permission PermissionName="pdrl-ex:com.adobe.aps.policySwitch"
Access="ALLOW"/>
      <Permission PermissionName="pdrl-ex:com.adobe.aps.revoke"
Access="ALLOW"/>
      <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.printHigh"
Access="ALLOW"/>
      <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.printLow"
Access="ALLOW"/>
      <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.copy"
Access="ALLOW"/>
      <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.edit"
Access="ALLOW"/>
      <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.editNotes"
Access="ALLOW"/>
      <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.fillAndSign"
Access="ALLOW"/>
      <Principal PrincipalNameType="SYSTEM">
         <PrincipalDomain>EDC_SPECIAL</PrincipalDomain>
         <PrincipalName>publisher</PrincipalName>
      </Principal>
   </PolicyEntry>
   <PolicyEntry>
      <Permission PermissionName="pdrl-ex:com.adobe.aps.onlineOpen"
Access="ALLOW"/>
```

```
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.offlineOpen"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.policySwitch"
Access="ALLOW"/>
     <Permission PermissionName="pdrl-ex:com.adobe.aps.revoke"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.printHigh"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.printLow"
Access="ALLOW"/>
     <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.copy"
Access="ALLOW"/>
     <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.edit"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.editNotes"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.fillAndSign"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.accessible"
Access="ALLOW"/>
     <PolicyEntryValidityPeriod isAbsoluteTime="true">
       <ValidityPeriodAbsolute>

<NotBeforeAbsolute>2004-06-04T10:00:00+00:00</NotBeforeAbsolute>

<NotAfterAbsolute>2004-07-05T10:00:00+00:00</NotAfterAbsolute>
       </ValidityPeriodAbsolute>
     </PolicyEntryValidityPeriod>
     <Principal PrincipalNameType="USER">
       <PrincipalDomain>adobe.com</PrincipalDomain>

<PrincipalName>uid=jsanfili,ou=people,o=adobe.com</PrincipalName>
     </Principal>
  </PolicyEntry>
  <PolicyEntry>
     <Permission PermissionName="pdrl-ex:com.adobe.aps.onlineOpen"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.offlineOpen"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.accessible"
Access="ALLOW"/>
```

```
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.printHigh"
Access="ALLOW"/>
     <Principal PrincipalNameType="USER">
       <PrincipalDomain>adobe.com</PrincipalDomain>

<PrincipalName>uid=jpravetz,ou=people,o=adobe.com</PrincipalName>
     </Principal>
     <Principal PrincipalNameType="USER">
       <PrincipalDomain>adobe.com</PrincipalDomain>

<PrincipalName>uid=jsanfili,ou=people,o=adobe.com</PrincipalName>
     </Principal>
   </PolicyEntry>
   <PolicyEntry>
     <Permission PermissionName="pdrl-ex:com.adobe.aps.onlineOpen"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.offlineOpen"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.accessible"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.printLow"
Access="ALLOW"/>
     <Permission
PermissionName="pdrl-ex:com.adobe.aps.pdf.fillAndSign"
Access="ALLOW"/>
     <Principal PrincipalNameType="GROUP">
       <PrincipalDomain>adobe.com</PrincipalDomain>
       <PrincipalName>cn=jsanfili-direct
reports,ou=groups,o=adobe.com</PrincipalName>
     </Principal>
   </PolicyEntry>
   <Watermark isWatermarked="true">
     <TemplateID>FEF70094-447F-07C5-EC13-01A6BEC4C2CC
</TemplateID>
   </Watermark>
   <pdrl-ex:AcrobatCondition>
     <pdrl-ex:PlaintextMetadata>true</pdrl-ex:PlaintextMetadata>

<pdrl-ex:EncryptFileAttachmentOnly>false</pdrl-ex:EncryptFileAttac
hmentOnly>
   </pdrl-ex:AcrobatCondition>
   <PolicyValidityPeriod isAbsoluteTime="false">
     <ValidityPeriodRelative>
       <NotAfterRelative>P30D</NotAfterRelative>
     </ValidityPeriodRelative>
   </PolicyValidityPeriod>
   <OfflineLeasePeriod>
```

```
      <Duration>P3D</Duration>
   </OfflineLeasePeriod>
   <Property PropertyName="DocumentumProperty1">
      <PropertyValue>value1</PropertyValue>
      <PropertyValue>value2</PropertyValue>
   </Property>
</Policy>
```

Each part of this sample policy is discussed below to highlight some of the key features:

- **Lines 6-65:** The policy contains four policy entries. A policy can contain zero or more policy entries.
- **Lines 6-21:** This is a single policy entry. It lists a single principal, which is a special SYSTEM principal that represents the document "publisher." This principal is granted the permissions to open the document online and offline, switch the policy, print in high resolution, copy, edit, etc. "Publisher" gets bound to the principal defined within the resource object at the time of document publishing.
- **Lines 22-44:** This policy entry lists a USER, "jsanfili." This principal is granted various permissions. The interesting difference in this policy entry is the policy entry validity period. This user is only granted the listed permissions between 6/4/2004 and 7/5/2004. The validity period at the policy level takes precedence. The policy entry validity period must be within the window of time granted by the policy level validity period. See Policy evaluation for the policy evaluation algorithm.
- **Lines 46-58:** This is an example of a policy entry that contains multiple principals. Both "jsanfili" and "jpravetz" are granted the listed permissions. Permissions are additive. A user can appear in multiple policy entries, and the union of all permissions is used to determine rights for a principal. See Policy evaluation for the policy evaluation algorithm.
- **Lines 59-69:** This is an example of a policy entry that contains a GROUP principal. Any user within that group will inherit the list permissions. Group membership is controlled by the external enterprise directory server so that membership changes can be dynamically changed per policy evaluation.
- **Lines 70-77:** These are examples of policy conditions. These conditions are very Acrobat-specific. They determine whether Rights Management 10 should audit user interactions with the document, whether a dynamic watermark should be applied to the document, and how to process metadata and file attachments. The behavior of Acrobat based on these conditions is defined outside the scope of this document.
- **Lines 78-82:** These lines define the validity period for the entire document. In this example the validity period is relative to the publish date, and the document will expire 30 days after being published. No users will have access to the document after the validity period has expired.
- **Lines 83-85:** These lines define the offline lease period. It is a condition which defines how long the document can be viewed offline before the user has to reconnect to the server and refresh the lease.
- **Lines 86-89:** This is an example of adding properties (name/value pairs) that can be used by a third-party evaluation engine.

Policy evaluation

PDRL does not define the policy evaluation algorithm. Since PDRL can be used in multiple policy evaluation systems, it is the responsibility of the system to define the evaluation rules.

For Rights Management 10, the following policy evaluation algorithm is implemented.

```
boolean evaluateValidityPeriod(ValidityPeriod vp,
                               Date evalTime,
                               Date publishTime) {
   if (vp.isAbsolute)
   return whether evalTime is within the absolute range of vp
   if (!vp.isAbsolute && publishTime != null )
   return whether evalTime is within range relative to publishTime
   return false;
}

// returns a set of permissions
Set evaluate(Principal userToEvaluate,
             Policy policy,
             Date evalTime,
             Date publishTime){
   // publishtime can be null if evaluating a policy without a
license
   Set relevantPrincipals = getGroupsForUser(user) union user;

   Set returnedPermissions = null;

   if ( evalTime not within policy.getValidity() )
     throw Exception("Policy expired.");

   foreach policyEntry in policy {
     if (!(Date.now() within policyEntry.getValidity())) continue;

     Set policyEntryPrincipals = policyEntry.getPrincipals();
     if (isEmpty(policyEntryPrincipals intersect
relevantPrincipals))
        continue;

     foreach permission in policyEntry.getPermissions {
        /* "permission" has name of the right and whether it is
           granted or denied */
        add permission to returnedPermissions;
     }
   }
   foreach granted permission in returnedPermissions{
     /* If a permission has been both granted and denied, it is
denied. */
     if(denied permission in returnedPermissions)remove permission
from returnedPermissions;
```

```
    }

    return returnedPermissions;
}
```

3 Extensibility

There are several areas in which the PDRL schema can be extended. A basic set of building blocks are provided within the core schema, which can be extended with implementation-specific rights and data.

The primary elements that allow for extensibility are *policy* and *license*.

Policy extensibility

The policy schema can be extended through *permissions*, *conditions*, and *properties*, as described in the following sections.

Permissions

New *permissions* can be added through additional XML schema namespaces. The following example demonstrates how permissions are defined for use by Rights Management 10.

The core schema contains this definition for a permission type. The extensibility point is the permission name itself which is simply a QName.

```
<complexType name="PermissionType">
   <annotation>
      <documentation>Type Definition: A permission to perform a document
operation.</documentation>
   </annotation>
   <attribute name="PermissionName" type="QName" use="required"/>
   <attribute name="Access" type="pdrl:PermissionAccessType" use="required"/>
</complexType>
```

A separate schema, "pdrl-ex.xsd", contains the permission names that are specific to Rights Management 10.

```
<schema targetNamespace="http://www.adobe.com/schema/1.0/pdrl-ex"
xmlns="http://www.w3.org/2001/XMLSchema"xmlns:pdrl="http://www.adobe.com/
schema/1.0/pdrl" xmlns:pdrl-ex="http://www.adobe.com/schema/1.0/pdrl-ex"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<simpleType name="PermissionNameEnum">
     <restriction base="QName">
        <enumeration value="pdrl-ex:com.adobe.aps.onlineOpen"/>
        <enumeration value="pdrl-ex:com.adobe.aps.offlineOpen"/>
        <enumeration value="pdrl-ex:com.adobe.aps.revoke"/>
        <enumeration value="pdrl-ex:com.adobe.aps.policySwitch"/>
        <enumeration value="pdrl-ex:com.adobe.aps.pdf.printHigh"/>
        <enumeration value="pdrl-ex:com.adobe.aps.pdf.printLow"/>
                    …
     </restriction>
   </simpleType>
</schema>
```

Other permission namespaces should be defined in a similar manner.

To add a custom permission, specify the namespace, which should be unique and is typically a domain name, followed by a colon, followed by the name of the permission. The namespace and permission name are assigned to the `PermissionName` attribute. For example, the following line specifies a custom permission whose namespace is `www.adele.com` and whose permission name is `DO_ANYTHING_YOU_WANT`:

```
   <pdrl:Permission
PermissionName="www.adele.com:DO_ANYTHING_YOU_WANT" Access="ALLOW"
/>
```

The following examples illustrate the addition of custom permissions.

Example: *Adding a custom permission*

```
<?xml version=”1.0” encoding=”UTF-8”?>
<pdrl:Policy
  xmlns:pdrl=”http://www.adobe.com/schema/1.0/pdrl”
  xmlns:pdrl-ex=”http://www.adobe.com/schema/1.0/pdrl-ex”
  xmlns:www.adele.com=”http://www.adele.com”
  PolicyID=””
  PolicyInstanceVersion=”0”
  PolicySchemaVersion=””
  PolicyName=”USER_POLICY:Wed Jan 17 16:53:42 2007 8027”

PolicyDescription=”TestPDRLPolicyEntryFunc:addPermission/getPermis
sions:”
  PolicyType=”2”
>
   <pdrl:PolicyEntry>
     <pdrl:Permission
       PermissionName=”www.adele.com:DO_ANYTHING_YOU_WANT”
       Access=”ALLOW”
     >
     </pdrl:Permission>
     <pdrl:Principal PrincipalNameType=”SYSTEM”>
       <pdrl:PrincipalDomain>EDC_SPECIAL</pdrl:PrincipalDomain>
       <pdrl:PrincipalName>publisher</pdrl:PrincipalName>
     </pdrl:Principal>
   </pdrl:PolicyEntry>
  <pdrl:Property PropertyName=”isCertified”>
    <pdrl:PropertyValue>false</pdrl:PropertyValue>
  </pdrl:Property>
  <pdrl:AuditSettings isTracked=”true”></pdrl:AuditSettings>
  <pdrl:Watermark isWatermarked=”false”></pdrl:Watermark>
  <pdrl-ex:AcrobatCondition>
    <pdrl-ex:PlaintextMetadata>false</pdrl-ex:PlaintextMetadata>

<pdrl-ex:EncryptFileAttachmentOnly>false</pdrl-ex:EncryptFileAttac
hmentOnly>
  </pdrl-ex:AcrobatCondition>
</pdrl:Policy>
```

Example: *Adding multiple custom permissions*

In this example, several custom permissions are added to the policy.

```
<?xml version="1.0" encoding="UTF-8"?>
<pdrl:Policy
  xmlns:pdrl="http://www.adobe.com/schema/1.0/pdrl"
  xmlns:pdrl-ex="http://www.adobe.com/schema/1.0/pdrl-ex"
  xmlns:www.adele.com="http://www.adele.com"
  PolicyID=""
  PolicyInstanceVersion="0"
  PolicySchemaVersion=""
  PolicyName="USER_POLICY:Wed Jan 17 16:53:43 2007 8817"

PolicyDescription="TestPDRLPolicyEntryFunc:addPermission/getPermis
sions:"
  PolicyType="2"
>
  <pdrl:PolicyEntry>
    <pdrl:Permission
      PermissionName="www.adele.com:DO_ANYTHING_YOU_WANT"
      Access="ALLOW"
    />
    <pdrl:Permission
      PermissionName="pdrl-ex:com.adobe.aps.offlineOpen"
      Access="ALLOW"
    />
    <pdrl:Permission
      PermissionName="pdrl-ex:com.adobe.aps.onlineOpen"
      Access="ALLOW"
    />
    <pdrl:Permission
      PermissionName="pdrl-ex:com.adobe.aps.pdf.accessible"
      Access="ALLOW"
    />
    <pdrl:Permission
      PermissionName="pdrl-ex:com.adobe.aps.pdf.copy"
      Access="ALLOW"
    />
    <pdrl:Permission
      PermissionName="pdrl-ex:com.adobe.aps.pdf.fillAndSign"
      Access="ALLOW"
    />
    <pdrl:Permission
      PermissionName="pdrl-ex:com.adobe.aps.pdf.printHigh"
      Access="ALLOW"
    />
    <pdrl:Permission
      PermissionName="pdrl-ex:com.adobe.aps.pdf.printLow"
```

```
      Access="ALLOW"
    />
    <pdrl:Permission
      PermissionName="pdrl-ex:com.adobe.aps.policySwitch"
      Access="ALLOW"
    />
    <pdrl:Permission
      PermissionName="pdrl-ex:com.adobe.aps.revoke"
      Access="ALLOW"
    />
    <pdrl:Principal PrincipalNameType="SYSTEM">
      <pdrl:PrincipalDomain>EDC_SPECIAL</pdrl:PrincipalDomain>
      <pdrl:PrincipalName>publisher</pdrl:PrincipalName>
    </pdrl:Principal>
  </pdrl:PolicyEntry>
  <pdrl:Property PropertyName="isCertified">
    <pdrl:PropertyValue>false</pdrl:PropertyValue>
  </pdrl:Property>
  <pdrl:AuditSettings isTracked="true"></pdrl:AuditSettings>
  <pdrl:Watermark isWatermarked="false"></pdrl:Watermark>
  <pdrl-ex:AcrobatCondition>
    <pdrl-ex:PlaintextMetadata>false</pdrl-ex:PlaintextMetadata>

<pdrl-ex:EncryptFileAttachmentOnly>false</pdrl-ex:EncryptFileAttac
hmentOnly>
  </pdrl-ex:AcrobatCondition>
</pdrl:Policy>
```

Example: *Adding multiple custom permissions*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Policy
  PolicyCreationTime="2007-03-16T20:03:47.375+00:00"
  PolicyDescription=""
  PolicyID="01E9A585-F153-418F-67B5-DC8CEE1C6705"
  PolicyInstanceVersion="3"
  PolicyName="Test"
  PolicySchemaVersion="1.0"
  PolicyType="2"
  xmlns="http://www.adobe.com/schema/1.0/pdrl"
>
  <AuditSettings isTracked="true"/>
  <PolicyEntry>
    <ns1:Permission
      Access="ALLOW"
      PermissionName="com.adobe.aps.onlineOpen"
      xmlns:ns1="http://www.adobe.com/schema/1.0/pdrl"
      xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
    />
    <ns2:Permission
```

```
  Access="ALLOW"
  PermissionName="com.adobe.aps.offlineOpen"
  xmlns:ns2="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns3:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.policySwitch"
  xmlns:ns3="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns4:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.revoke"
  xmlns:ns4="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns5:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.pdf.printHigh"
  xmlns:ns5="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns6:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.pdf.printLow"
  xmlns:ns6="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns7:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.pdf.copy"
  xmlns:ns7="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns8:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.pdf.accessible"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
  xmlns:ns8="http://www.adobe.com/schema/1.0/pdrl"
/>
<ns9:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.pdf.edit"
  xmlns:ns9="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns10:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.pdf.editNotes"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
```

```
      xmlns:ns10="http://www.adobe.com/schema/1.0/pdrl"
    />
    <ns11:Permission
      Access="ALLOW"
      PermissionName="com.adobe.aps.pdf.fillAndSign"
      xmlns:ns11="http://www.adobe.com/schema/1.0/pdrl"
      xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
    />
    <Principal PrincipalNameType="SYSTEM">
      <PrincipalDomain>EDC_SPECIAL</PrincipalDomain>
      <PrincipalName>publisher</PrincipalName>
    </Principal>
  </PolicyEntry>
  <ns12:AcrobatCondition
xmlns:ns12="http://www.adobe.com/schema/1.0/pdrl-ex">
    <ns12:PlaintextMetadata>false</ns12:PlaintextMetadata>

<ns12:EncryptFileAttachmentOnly>false</ns12:EncryptFileAttachmentO
nly>
  </ns12:AcrobatCondition>
  <Watermark isWatermarked="false"/>
  <OfflineLeasePeriod>
    <Duration>P30D</Duration>
  </OfflineLeasePeriod>
  <Property PropertyName="external authorizer">
    <PropertyValue>PrintControlSPISample</PropertyValue>
  </Property>
  <Property PropertyName="isCertified">
    <PropertyValue>false</PropertyValue>
  </Property>
</Policy>
```

Conditions

There are four types of conditions. *Conditions* can apply either to the entire policy (for example, whether the actions taken by a principal using this policy should be audited) or to a particular policy entry within the policy (for example, a validity period that limits the use

of a policy entry). Conditions are all derived from the abstract complex type `ConditionAbstractType`:

`PolicyEntryConditionSingle`: Conditions that are specific to policy entries. Conditions of this type should set their substitution group attribute to this element. Use for 0..1 cardinality conditions.

`PolicyEntryConditionMulti`: Conditions that are specific to policy entries. Conditions of this type should set their substitution group attribute to this element. Use for 0..n cardinality conditions.

`PolicyConditionSingle`: Conditions that are specific to the entire policy. Conditions of this type should set their substitution group attribute to this element. Use for 0..1 cardinality conditions.

`PolicyConditionMulti`: Conditions that are specific to the entire policy. Conditions of this type should set their substitution group attribute to this element. Use for 0..n cardinality conditions.

The following XML shows a condition used for policy entries.

```
<element name="PolicyEntryValidityPeriod"
type="pdrl:ValidityPeriodType"
substitutionGroup="pdrl:PolicyEntryConditionMulti">
     <annotation>
       <documentation> PolicyEntry Condition: The time period
between which an object is valid. It consists of the time not before
and not after. It can be absolute or relative. If relative, it is
relative to the License creation time.</documentation>
     </annotation>
</element>
  <element name="PolicyValidityPeriod"
type="pdrl:ValidityPeriodType"
substitutionGroup="pdrl:PolicyConditionSingle">
     <annotation>
       <documentation> Policy Condition: The time period between
which an object is valid. It consists of the time not before and not
after. It can be absolute or relative. If relative, it is relative
to the License creation time.</documentation>
     </annotation>
  </element>
```

The following XML shows a condition that applies to the entire policy.

```
<element name="AuditSettings" type="pdrl:AuditSettingsType"
substitutionGroup="pdrl:PolicyConditionSingle">
  <annotation>
    <documentation>  Policy Condition: Audit settings. Is the
document tracked. boolean.</documentation>
  </annotation>
</element>
```

Properties

A *property* element is a simple name/value pair container. The property value can be single or multi-valued. The type of the value can be any simple data type (integer, string, Boolean, …), which allows for simple extensibility through the addition of application-specific name/value pairs to the policy. These properties are application-specific and typically used to customize the policy evaluation process with policy instance-specific data.

Here is the definition of a property type:

```
<complexType name="PropertyType">
   <annotation>
     <documentation>Type Definition: Property is a simple
name/value pair container. It can be single or multi-valued. The
type can be any simple data type.</documentation>
   </annotation>
   <sequence>
     <element name="PropertyValue" type="anySimpleType"
maxOccurs="unbounded"/>
   </sequence>
   <attribute name="PropertyName" type="string" use="required"/>
   <attribute name="PropertyNamespace" type="string"
use="optional"/>
</complexType>
```

The following XML shows how a property can be used within a policy.

```
<Property PropertyName="DocumentumProperty1">
   <PropertyValue>value1</PropertyValue>
   <PropertyValue>value2</PropertyValue></Property>
```

To use the external authorization SPI, set the `PropertyName` attribute to `"external authorizer"` and set the `PropertyValue` element to the name of the registered external authorization provider, as shown in the following example.

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<Policy
  PolicyCreationTime=”2007-03-16T20:03:47.375+00:00”
  PolicyDescription=””
  PolicyID=”01E9A585-F153-418F-67B5-DC8CEE1C6705”
  PolicyInstanceVersion=”3”
  PolicyName=”Test”
  PolicySchemaVersion=”1.0”
  PolicyType=”2”
  xmlns=”http://www.adobe.com/schema/1.0/pdrl”
>
  <AuditSettings isTracked=”true”/>
  <PolicyEntry>
    <ns1:Permission
      Access=”ALLOW”
      PermissionName=”com.adobe.aps.onlineOpen”
      xmlns:ns1=”http://www.adobe.com/schema/1.0/pdrl”
```

```
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns2:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.offlineOpen"
  xmlns:ns2="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns3:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.policySwitch"
  xmlns:ns3="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns4:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.revoke"
  xmlns:ns4="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns5:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.pdf.printHigh"
  xmlns:ns5="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns6:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.pdf.printLow"
  xmlns:ns6="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns7:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.pdf.copy"
  xmlns:ns7="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns8:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.pdf.accessible"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
  xmlns:ns8="http://www.adobe.com/schema/1.0/pdrl"
/>
<ns9:Permission
  Access="ALLOW"
  PermissionName="com.adobe.aps.pdf.edit"
  xmlns:ns9="http://www.adobe.com/schema/1.0/pdrl"
  xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
/>
<ns10:Permission
```

```
      Access="ALLOW"
      PermissionName="com.adobe.aps.pdf.editNotes"
      xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
      xmlns:ns10="http://www.adobe.com/schema/1.0/pdrl"
    />
    <ns11:Permission
      Access="ALLOW"
      PermissionName="com.adobe.aps.pdf.fillAndSign"
      xmlns:ns11="http://www.adobe.com/schema/1.0/pdrl"
      xmlns="http://www.adobe.com/schema/1.0/pdrl-ex"
    />
    <Principal PrincipalNameType="SYSTEM">
      <PrincipalDomain>EDC_SPECIAL</PrincipalDomain>
      <PrincipalName>publisher</PrincipalName>
    </Principal>
  </PolicyEntry>
  <ns12:AcrobatCondition
xmlns:ns12="http://www.adobe.com/schema/1.0/pdrl-ex">
    <ns12:PlaintextMetadata>false</ns12:PlaintextMetadata>

<ns12:EncryptFileAttachmentOnly>false</ns12:EncryptFileAttachmentO
nly>
  </ns12:AcrobatCondition>
  <Watermark isWatermarked="false"/>
  <OfflineLeasePeriod>
    <Duration>P30D</Duration>
  </OfflineLeasePeriod>
  <Property PropertyName="external authorizer">
    <PropertyValue>PrintControlSPISample</PropertyValue>
  </Property>
  <Property PropertyName="isCertified">
    <PropertyValue>false</PropertyValue>
  </Property>
</Policy>
```

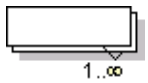
License extensibility

The license can be extended through properties. This type definition is the same one described in Properties.

4 Core Schema

This chapter defines the details of the core PDRPolicyIDL schema. The schema is described using diagrams created with Altova's XML Spy, which uses the following symbols.

The cardinality of the element (0..1, 1 exactly, 0..n, 1..n) is indicated by the border of the elements. Optional elements are drawn with a dashed line and required elements with a solid line. A maximum occurrence greater than one is indicated by a double border.

		
Optional element Min. occurrence = 0, Max. occurrence = 1	Required single element Min. occurrence = 1, Max. occurrence = 1	Required repeated element Min. occurrence = 1, Max. occurrence = unbounded

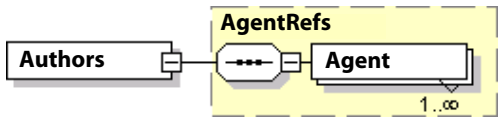
The content model of elements is symbolized on the left and right side of the element boxes. The left side indicates whether the element contains a simple type (text, numbers, dates, etc.) or a complex type (further elements). The right side of the element symbol indicates whether it contains child elements.

			
simple content	complex content	complex content with child elements	no element content (simple type, attributes only, or empty element)

Other symbols include these.

	A sequence of elements. The elements must appear exactly in the sequence in which they appear in the schema diagram.
	A choice of elements. Only a single element from those in the choice may appear at this position.
	The "all" model, in which the sequence of elements is not fixed.

Finally, if an element refers to a complex global type, the type is shown with a border and yellow background.



License element

License is the primary container for rights information. It consists of the *issuing authority*, *resource*, *policy*, and *signature*. It is a binding of policy to a resource that determines rights to the resource.

The License element contains the following attributes:

`LicenseID`: The unique identifier of the license instance. It is optional when the instance is created, but the server must fill in a value before it can be used in a security workflow.

`LicenseInstanceVersion`: A counter that gets incremented each time a change is made to the license data. It is optional when the instance is created, but the server must fill in a value before it can be used in a security workflow.

`LicenseIssueTime`: The time that the License was issued. It is optional when the instance is created, but the server must fill in a value before it can be used in a security workflow.

`LicenseSchemaVersion`: The version of the schema used by this instance. It is optional when the instance is created, but the server must fill in a value before it can be used in a security workflow.

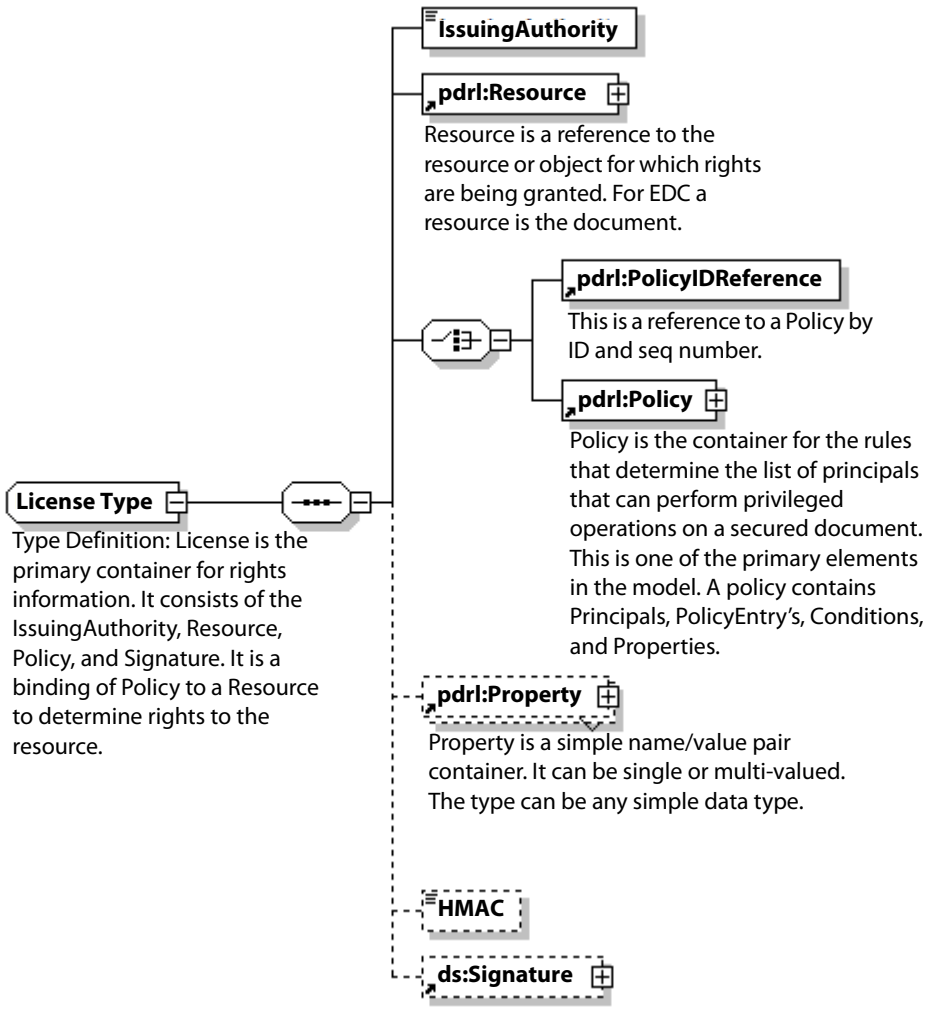
element License

source

```
<element name="License" type="pdrl:LicenseType">
   <annotation>
      <documentation> License is the primary container for rights
information.
      It consists of the IssuingAuthority, Resource, Policy, and
Signature. It
      is a binding of Policy to a Resource to determine rights to the
      resource.</documentation>
   </annotation>
</element>
```

complexType LicenseType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

children

`IssuingAuthority pdrl:Resource pdrl:PolicyIDReference pdrl:Policy pdrl:Property HMAC ds:Signature`

used by

`element License`

attributes

Name	Type	Use
LicenseID	string	optional
LicenseInstanceVersion	int	optional
LicenseIssueTime	dateTime	optional
LicenseSchemaVersion	string	optional

source

```
<complexType name="LicenseType">
   <annotation>
      <documentation> Type Definition:  License is the primary
container
      for rights information. It consists of the IssuingAuthority,
      Resource, Policy, and Signature. It is a binding of Policy to a
      Resource to determine rights to the resource.</documentation>
   </annotation>
   <sequence>
      <element name="IssuingAuthority" type="anyURI"/>
      <element ref="pdrl:Resource"/>
      <choice>
         <element ref="pdrl:PolicyIDReference"/>
         <element ref="pdrl:Policy"/>
      </choice>
      <element ref="pdrl:Property" minOccurs="0"
maxOccurs="unbounded"/>
      <element name="HMAC" type="pdrl:HMACType" minOccurs="0"/>
      <element ref="ds:Signature" minOccurs="0"/>
   </sequence>
   <attribute name="LicenseID" type="string" use="optional"/>
   <attribute name="LicenseInstanceVersion" type="int"
use="optional"/>
   <attribute name="LicenseIssueTime" type="dateTime"
use="optional"/>
   <attribute name="LicenseSchemaVersion" type="string"
use="optional"/>
</complexType>
```

IssuingAuthority element

IssuingAuthority is the identifier for the trusted security domain that issued the license. It is represented as a URI.

element LicenseType/IssuingAuthority

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

anyURI

source

```
<element name="IssuingAuthority" type="anyURI"/>
```

PolicyIDReference element

PolicyIDReference is a reference to a policy by `PolicyID`. `PolicyID` must be a unique identifier.

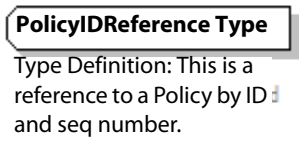
element PolicyIDReference

source

```
<element name="PolicyIDReference"
type="pdrl:PolicyIDReferenceType">
   <annotation>
      <documentation> This is a reference to a Policy by
ID.</documentation>
   </annotation>
</element>
```

complexType PolicyIDReferenceType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

used by

`element PolicyIDReference`

attributes

Name	Type	Use
`PolicyID`	`string`	required

source

```
<complexType name="PolicyIDReferenceType">
   <annotation>
```

```
      <documentation> Type Definition:  This is a reference to a
Policy
      by ID </documentation>
   </annotation>
   <attribute name="PolicyID" type="string" use="required"/>
</complexType>
```

Property element

Property is a simple name/value pair container. The property can be single or multi-valued. The type of the values can be any simple data type. This is an extensibility point to allow third parties to include application-specific data within licenses and policies.

Attributes:

`PropertyName`: The name of the property.

`PropertyNamespace`: Optional attribute and can be used to group "like" properties into a single category.

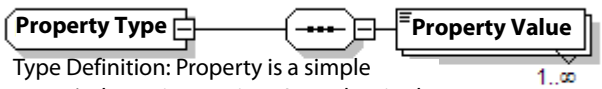
element Property

source

```
<element name="Property" type="pdrl:PropertyType">
   <annotation>
      <documentation>Property is a simple name/value pair container.
      It can be single or multi-valued. The type can be any simple
data
      type.</documentation>
   </annotation>
</element>
```

ComplexType PropertyType

diagram



Type Definition: Property is a simple name/value pair container. It can be single or multi-valued. The type can be any simple data type.

namespace

http://www.adobe.com/schema/1.0/pdrl

children

PropertyValue

used by

element Property

attributes

Name	Type	Use
PropertyName	string	required
PropertyNamespace	string	optional

source

```
<complexType name="PropertyType">
   <annotation>
      <documentation>Type Definition: Property is a simple
name/value pair
      container. It can be single or multi-valued. The type can be
any
      simple data type.</documentation>
   </annotation>
   <sequence>
      <element name="PropertyValue" type="anySimpleType"
      maxOccurs="unbounded"/>
   </sequence>
   <attribute name="PropertyName" type="string" use="required"/>
   <attribute name="PropertyNamespace" type="string"
use="optional"/></complexType>
```

element PropertyType/PropertyValue

The value of a property. This value can be any data type. Note that multi-valued properties are supported.

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

anySimpleType

source

```
<element name="PropertyValue" type="anySimpleType"
maxOccurs="unbounded"/>
```

HMAC element

HMAC is used to provide integrity protection to the license. The algorithm for creating the HMAC is defined in RFC 2104. The HMAC is Base64 encoded. The key used to create the HMAC is stored separately from the license. Note that either an HMAC or an XML digital signature can be used to protect the integrity of the license.

element LicenseType/HMAC

source

```
<element name="HMAC" type="pdrl:HMACType" minOccurs="0"/>
```

simpleType HMACType

namespace

http://www.adobe.com/schema/1.0/pdrl

type

```
base64Binary
```

used by

```
element LicenseType/HMAC
```

source

```
<simpleType name="HMACType">
  <annotation>
    <documentation> Type Definition: Stores the HMAC value as a
Base64
    encoded octet string.</documentation>
  </annotation>
  <restriction base="base64Binary"/>
</simpleType>
```

ds:Signature element

ds:Signature is used to provide integrity protection to the license. The algorithm for creating the signature is defined in the XML digital signature specification [DSIG]. The private key used to create the signature is stored separately from the license. Note that either an HMAC or an XML digital signature can be used to integrity protect the license.

Policy element

Policy is the container for the rules that determine the list of principals that can perform privileged operations on a secured document. This is one of the primary elements in the model. A policy contains principals, policy entries, conditions, and properties.

Attributes:

`PolicyID`: The unique identifier of the policy instance. It is optional when the instance is created, but the server must fill in a value before it can be used in a security workflow.

`PolicyInstanceVersion`: A counter that gets incremented each time a change is made to the policy data. It is optional when the instance is created, but the server must fill in a value before it can be used in a security workflow.

`PolicyCreationTime`: The time that the policy was created. It is optional when the instance is created, but the server must fill in a value before it can be saved and used in a security workflow.

`PolicySchemaVersion`: The version of the schema used by this instance. It is optional when the instance is created, but the server must fill in a value before it can be used in a security workflow.

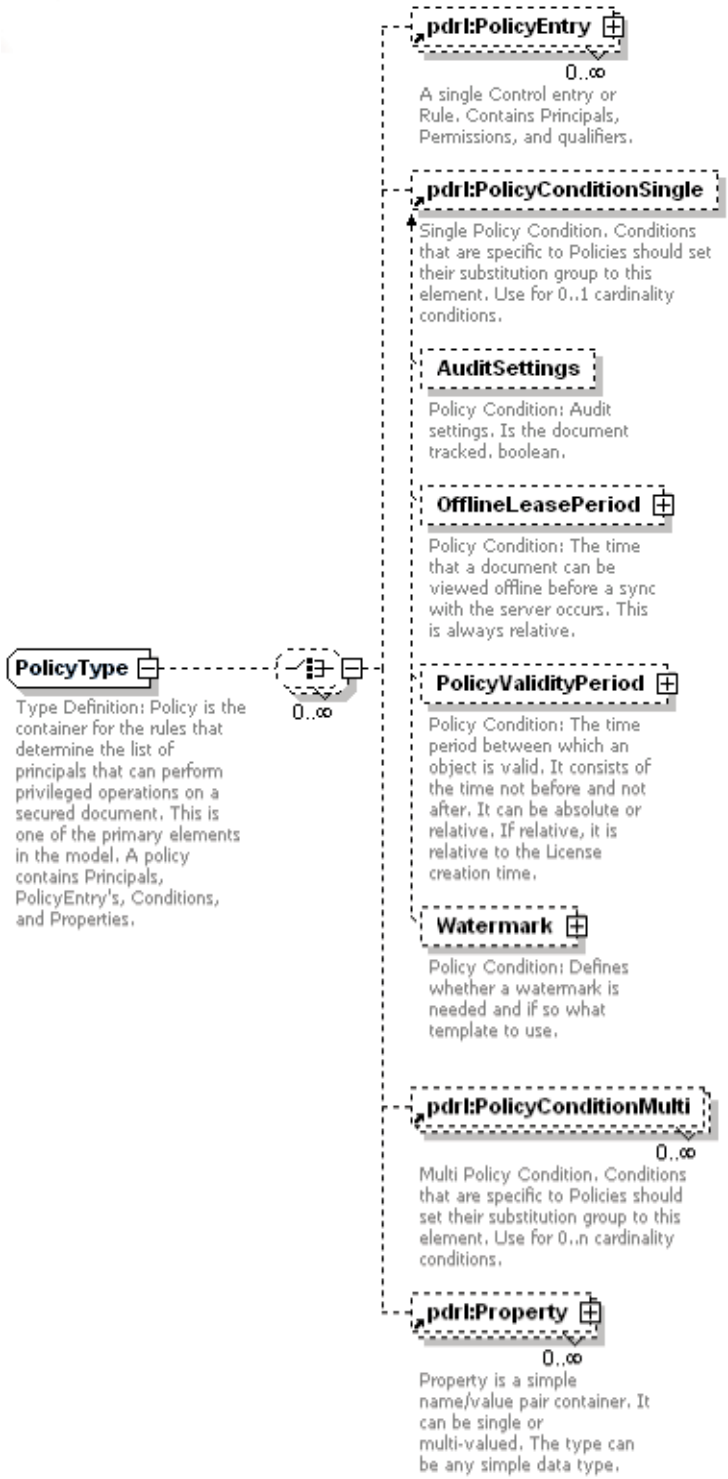
element Policy

source

```
<element name="Policy" type="pdrl:PolicyType">
   <annotation>
      <documentation>Policy is the container for the rules that
determine
      the list of principals that can perform privileged operations
on a
      secured document. This is one of the primary elements in the
model. A
      policy contains Principals, PolicyEntry's, Conditions, and
      Properties.</documentation>
   </annotation>
</element>
```

complexType PolicyType

diagram


PolicyType
Type Definition: Policy is the container for the rules that determine the list of principals that can perform privileged operations on a secured document. This is one of the primary elements in the model. A policy contains Principals, PolicyEntry's, Conditions, and Properties.
0..∞
pdrl:PolicyEntry
0..∞
A single Control entry or Rule. Contains Principals, Permissions, and qualifiers.
pdrl:PolicyConditionSingle
Single Policy Condition. Conditions that are specific to Policies should set their substitution group to this element. Use for 0..1 cardinality conditions.
AuditSettings
Policy Condition: Audit settings. Is the document tracked. boolean.
OfflineLeasePeriod
Policy Condition: The time that a document can be viewed offline before a sync with the server occurs. This is always relative.
PolicyValidityPeriod
Policy Condition: The time period between which an object is valid. It consists of the time not before and not after. It can be absolute or relative. If relative, it is relative to the License creation time.
Watermark
Policy Condition: Defines whether a watermark is needed and if so what template to use.
pdrl:PolicyConditionMulti
0..∞
Multi Policy Condition. Conditions that are specific to Policies should set their substitution group to this element. Use for 0..n cardinality conditions.
pdrl:Property
0..∞
Property is a simple name/value pair container. It can be single or multi-valued. The type can be any simple data type.

namespace

http://www.adobe.com/schema/1.0/pdrl

children

pdrl:PolicyEntry pdrl:PolicyConditionSingle
pdrl:PolicyConditionMulti pdrl:Property

used by

element Policy

attributes

Name	Type	Use
PolicyID	string	optional
PolicyInstanceVersion	int	optional
PolicySchemaVersion	string	optional
PolicyCreationTime	dateTime	optional

source

```
<complexType name="PolicyType">
   <annotation>
      <documentation>Type Definition: Policy is the container for
the rules
      that determine the list of principals that can perform
privileged
      operations on a secured document. This is one of the primary
elements
      in the model. A policy contains Principals, PolicyEntry's,
Conditions,
      and Properties.</documentation>
   </annotation>
   <choice minOccurs="0" maxOccurs="unbounded">
      <element ref="pdrl:PolicyEntry" minOccurs="0"
maxOccurs="unbounded"/>
      <element ref="pdrl:PolicyConditionSingle" minOccurs="0"/>
      <element ref="pdrl:PolicyConditionMulti" minOccurs="0"
      maxOccurs="unbounded"/>
      <element ref="pdrl:Property" minOccurs="0"
maxOccurs="unbounded"/>
   </choice>
   <attribute name="PolicyID" type="string" use="optional"/>
```

```
   <attribute name="PolicyInstanceVersion" type="int"
use="optional"/>
   <attribute name="PolicySchemaVersion" type="string"
use="optional"/>
   <attribute name="PolicyCreationTime" type="dateTime"
use="optional"/>
</complexType>
```

PolicyEntry element

This section describes a *PolicyEntry* element.

element PolicyEntry

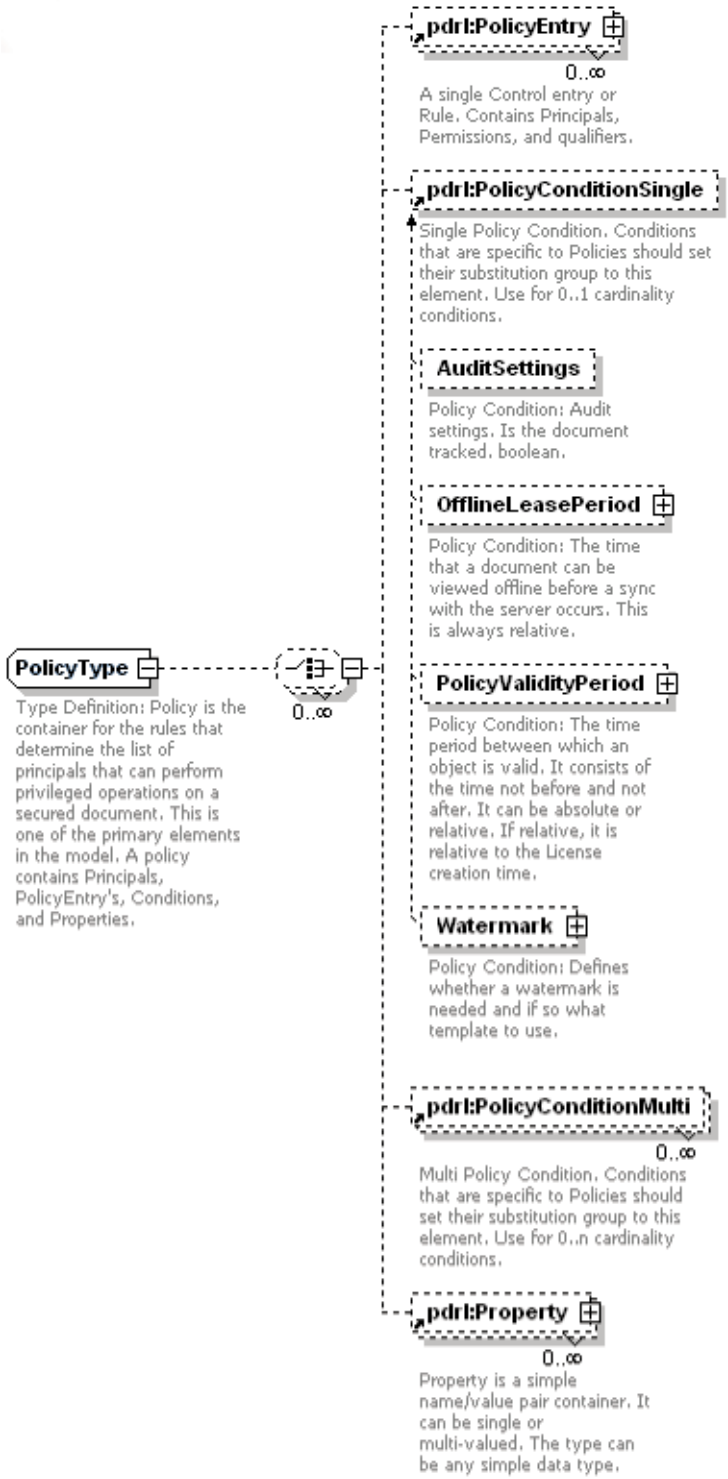
PolicyEntry is a single control entry or rule. It contains principals, permissions, and qualifiers. Zero or more entries may exist in the policy.

source

```
<element name="PolicyEntry" type="pdrl:PolicyEntryType">
  <annotation>
    <documentation>A single Control entry or Rule. Contains
Principals,
    Permissions, and qualifiers.</documentation>
  </annotation>
</element>
```

complexType PolicyEntryType

diagram


PolicyType
Type Definition: Policy is the container for the rules that determine the list of principals that can perform privileged operations on a secured document. This is one of the primary elements in the model. A policy contains Principals, PolicyEntry's, Conditions, and Properties.
0..∞
pdrl:PolicyEntry
0..∞
A single Control entry or Rule. Contains Principals, Permissions, and qualifiers.
pdrl:PolicyConditionSingle
Single Policy Condition. Conditions that are specific to Policies should set their substitution group to this element. Use for 0..1 cardinality conditions.
AuditSettings
Policy Condition: Audit settings. Is the document tracked. boolean.
OfflineLeasePeriod
Policy Condition: The time that a document can be viewed offline before a sync with the server occurs. This is always relative.
PolicyValidityPeriod
Policy Condition: The time period between which an object is valid. It consists of the time not before and not after. It can be absolute or relative. If relative, it is relative to the License creation time.
Watermark
Policy Condition: Defines whether a watermark is needed and if so what template to use.
pdrl:PolicyConditionMulti
0..∞
Multi Policy Condition. Conditions that are specific to Policies should set their substitution group to this element. Use for 0..n cardinality conditions.
pdrl:Property
0..∞
Property is a simple name/value pair container. It can be single or multi-valued. The type can be any simple data type.

namespace

http://www.adobe.com/schema/1.0/pdrl

children

```
pdrl:Principal pdrl:Permission pdrl:PolicyEntryConditionMulti
pdrl:PolicyEntryConditionSingle
```

used by

```
element PolicyEntry
```

source

```
<complexType name="PolicyEntryType">
   <annotation>
      <documentation>Type Definition: A single Control entry or
Rule.
      Contains Principals, Permissions, and
qualifiers.</documentation>
   </annotation>
   <choice minOccurs="0" maxOccurs="unbounded">
      <element ref="pdrl:Principal" maxOccurs="unbounded"/>
      <element ref="pdrl:Permission" maxOccurs="unbounded"/>
      <element ref="pdrl:PolicyEntryConditionMulti" minOccurs="0"
      maxOccurs="unbounded"/>
      <element ref="pdrl:PolicyEntryConditionSingle"
minOccurs="0"/>
   </choice>
</complexType>
```

Principal element

Principal contains the domain of the principal and the name of the principal. *PrincipalDomain* represents the user directory that stores the principal identities, and *PrincipalName* is the unique user ID within that user directory.

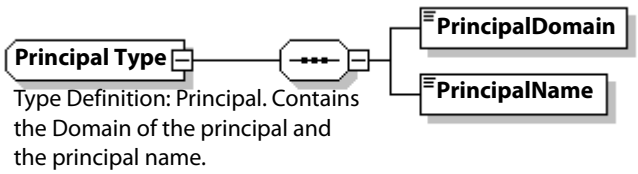
element Principal

source

```
<element name="Principal" type="pdrl:PrincipalType">
   <annotation>
      <documentation>Principal. Contains the Domain of the principal
      and the principal name.</documentation>
   </annotation>
</element>
```

complexType PrincipalType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

children

`PrincipalDomain PrincipalName`

used by

`elements Principal ResourceType/Publisher`

attributes

Name	Type	Use
`PrincipalNameType`	`pdrl:PrincipalNameTypeEnumeration`	required

source

```
<complexType name="PrincipalType">
  <annotation>
    <documentation>Type Definition: Principal. Contains the Domain
of the
    principal and the principal name.</documentation>
  </annotation>
  <sequence>
    <element name="PrincipalDomain" type="anyURI"/>
    <element name="PrincipalName" type="string"/>
  </sequence>
  <attribute name="PrincipalNameType"
  type="pdrl:PrincipalNameTypeEnumeration" use="required"/>
</complexType>
```

element PrincipalType/PrincipalDomain

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

anyURI

source

```
<element name="PrincipalDomain" type="anyURI"/>
```

element PrincipalType/PrincipalName

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

`string`

source

```
<element name="PrincipalName" type="string"/>
```

Note: A DN value assigned to the `PrincipalName` element must be in lowercase and not contain spaces. For example:

```
<PrincipalName>uid=jsanfili,ou=people,o=adobe.com</PrincipalName>
```

simpleType PrincipalNameTypeEnumeration

Enumeration values:

`USER`: A person.

`GROUP`: A group of people.

`ROLE`: A name representing a persons title or responsibility within an organization.

`SYSTEM`: A server machine.

`SERVICE`: A software component that exposes an external interface.

namespace

http://www.adobe.com/schema/1.0/pdrl

type

restriction of `string`

used by

`attribute PrincipalType/@PrincipalNameType`

facets

```
enumeration USER
enumeration GROUP
enumeration ROLE
enumeration SYSTEM
enumeration SERVICE
```

source

```
<simpleType name="PrincipalNameTypeEnumeration">
  <annotation>
```

```
      <documentation>Access qualifier for the Principal. It's a hint
of what
      type of principal this is "User", "Group", "Role", "System",
      "Service"</documentation>
    </annotation>
    <restriction base="string">
      <enumeration value="USER"/>
      <enumeration value="GROUP"/>
      <enumeration value="ROLE"/>
      <enumeration value="SYSTEM"/>
      <enumeration value="SERVICE"/>
    </restriction>
</simpleType>
```

Permission element

Permission represents a single right within a policy. It is an extensibility point. Application-specific permissions can be defined using this element. The permission is just a `QName` or string. The application needs to interpret that string and enforce a right based on the string.

Attributes:

`PermissionName`: The name of the permission.

`Access`: Is the permission enforcement algorithm supposed to `ALLOW` or `DENY` this right to the principal.

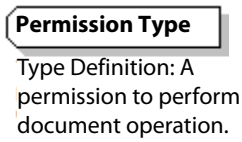
element Permission

source

```
<element name="Permission" type="pdrl:PermissionType">
   <annotation>
      <documentation> Permission Base class (abstract). Used as the
      extension base for permissions.</documentation>
   </annotation>
</element>
```

complexType PermissionType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

used by

`element Permission`

attributes

Name	Type	Use
PermissionName	QName	required
Access	pdrl:PermissionAccessType	required

source

```
<complexType name="PermissionType">
   <annotation>
      <documentation>Type Definition: A permission to perform a
document
      operation.</documentation>
   </annotation>
   <attribute name="PermissionName" type="QName" use="required"/>
   <attribute name="Access" type="pdrl:PermissionAccessType"
use="required"/>
</complexType>
```

simpleType PermissionAccessType

namespace

http://www.adobe.com/schema/1.0/pdrl

type

restriction of `string`

used by

```
attribute PermissionType/@Access
```

facets

```
enumeration ALLOW
enumeration DENY
```

source

```
<simpleType name="PermissionAccessType">
   <annotation>
      <documentation>Access qualifier for the PolicyEntry. Defines
the
      enumeration of "ALLOW" or "DENY"</documentation>
   </annotation>
```

```
      <restriction base="string">
        <enumeration value="ALLOW"/>
        <enumeration value="DENY"/>
      </restriction>
    </simpleType>
```

PolicyEntryCondition elements

Condition elements

Property element

Property element

Resource element

Resource is a reference to the resource or object for which rights are being granted. In this case, a resource is the document. Resource

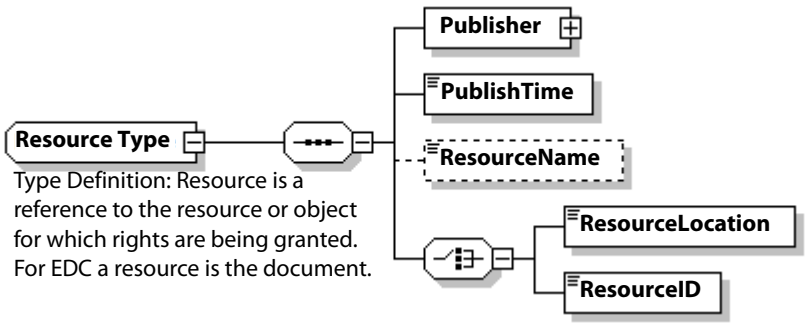
element Resource

source

```
<element name="Resource" type="pdrl:ResourceType">
  <annotation>
    <documentation> Resource is a reference to the resource or
object
    for which rights are being granted. In this case a resource is
the
    document.</documentation>
  </annotation>
</element>
```

complexType ResourceType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

children

Publisher PublishTime ResourceName ResourceLocation ResourceID

used by

element Resource

source

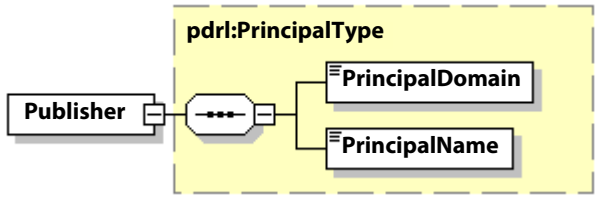
```
<complexType name="ResourceType">
  <annotation>
    <documentation> Type Definition:  Resource is a reference to
the
    resource or object for which rights are being granted. In this
case
    a resource is the document.</documentation>
  </annotation>
  <sequence>
    <element name="Publisher" type="pdrl:PrincipalType"/>
    <element name="PublishTime" type="dateTime"/>
    <element name="ResourceName" type="string" minOccurs="0"/>
    <choice>
      <element name="ResourceLocation" type="anyURI"/>
      <element name="ResourceID" type="string"/>
    </choice>
  </sequence>
</complexType>
```

Publisher element

Publisher is the creator or owner of a document.

element ResourceType/Publisher

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

```
pdrl:PrincipalType
```

children

```
PrincipalDomain PrincipalName
```

attributes

Name	Type	Use
PrincipalNameType	PrincipalNameTypeEnumeration	required

source

```
<element name="Publisher" type="pdrl:PrincipalType"/>
```

PublishTime element

PublishTime is the time that a resource was registered with the security system. In other words, the time when a resource instance object was created on the server.

element ResourceType/PublishTime

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

`dateTime`

source

```
<element name="PublishTime" type="dateTime"/>
```

ResourceName element

ResourceName is a friendly string to identify the resource. The string does not need to be unique.

element ResourceType/ResourceName

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

`String`

source

```
<element name="ResourceName" type="string" minOccurs="0"/>
```

ResourceLocation element

ResourceLocation is a possible way to uniquely identify a resource. It is a URI. If ResourceLocation is present, *ResourceID* is not present.

element ResourceType/ResourceLocation

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

anyURI

source

```
<element name="ResourceLocation" type="anyURI"/>
```

ResourceID element

ResourceID is a possible way to uniquely identify a resource. It is a URI. If ResourceID is present, *ResourceLocation* is not present.

element ResourceType/ResourceID

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

`String`

source

```
<element name="ResourceID" type="string"/>
```

Condition elements

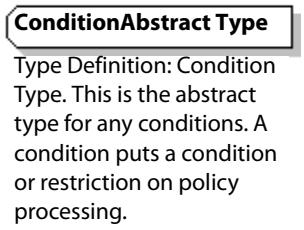
This section defines the condition schema. First, the base classes are defined. Then, the lists of conditions defined within the core schema are itemized.

ConditionAbstractType type

The abstract type for any conditions.

complexType ConditionAbstractType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

used by

```
elements PolicyConditionMulti PolicyConditionSingle
PolicyEntryConditionMulti PolicyEntryConditionSingle
complexTypes AuditSettingsType DeltaTimeType ValidityPeriodType
WatermarkType
```

source

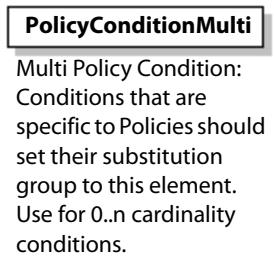
```
<complexType name="ConditionAbstractType" abstract="true">
  <annotation>
    <documentation> Type Definition: Condition Type. This is the
abstract
    type for any conditions. A condition puts a condition or
restriction on
    policy processing.</documentation>
  </annotation>
</complexType>
```

PolicyCondition and PolicyConditionEntry base elements

The following types are used as base classes for conditions within policies. See Conditions for more details on conditions.

element PolicyConditionMulti

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

`pdrl:ConditionAbstractType`

used by

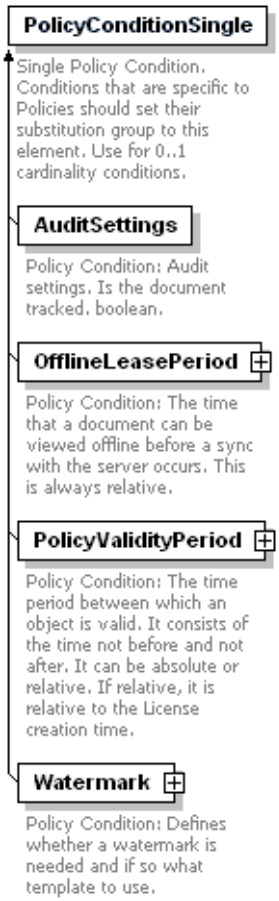
`complexType PolicyType`

source

```
<element name="PolicyConditionMulti"
type="pdrl:ConditionAbstractType">
   <annotation>
      <documentation> Multi Policy Condition. Conditions that are
specific
      to Policies should set their substitution group to this
element. Use
      for 0..n cardinality conditions.</documentation>
   </annotation>
</element>
```

element PolicyConditionSingle

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

`pdrl:ConditionAbstractType`

used by

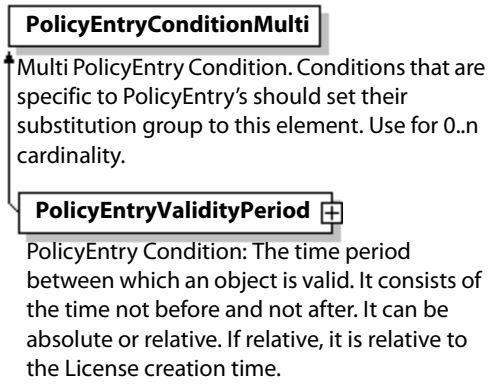
`complexType PolicyType`

source

```
<element name="PolicyConditionSingle"
type="pdrl:ConditionAbstractType">
  <annotation>
    <documentation> Single Policy Condition. Conditions that are
    specific to Policies should set their substitution group to
this
    element. Use for 0..1 cardinality conditions.</documentation>
  </annotation>
</element>
```

element PolicyEntryConditionMulti

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

```
pdrl:ConditionAbstractType
```

used by

```
complexType PolicyEntryType
```

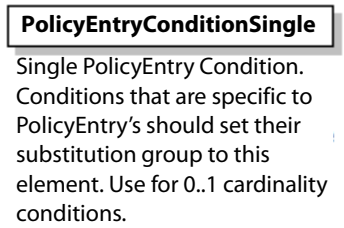
source

```
<element name="PolicyEntryConditionMulti"
type="pdrl:ConditionAbstractType">
  <annotation>
    <documentation> Multi PolicyEntry Condition. Conditions that
are
```

```
     specific to PolicyEntry's should set their substitution group
to
     this element. Use for 0..n cardinality.</documentation>
   </annotation>
</element>
```

element PolicyEntryConditionSingle

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

```
pdrl:ConditionAbstractType
```

used by

```
complexType PolicyEntryType
```

source

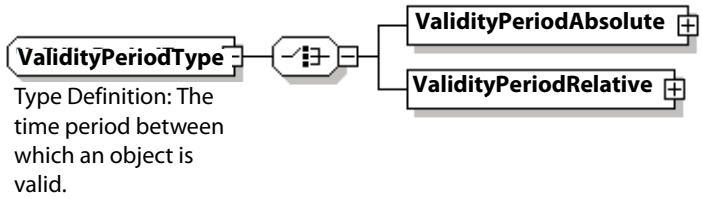
```
<element name="PolicyEntryConditionSingle"
type="pdrl:ConditionAbstractType">
   <annotation>
      <documentation>Single PolicyEntry Condition. Conditions that
are
      specific to PolicyEntry's should set their substitution group
to
      this element. Use for 0..1 cardinality
conditions.</documentation>
   </annotation>
</element>
```

ValidityPeriodType condition

ValidityPeriodType is a complex type used in the definition of time-related conditions. A validity period can be either absolute or relative. Absolute time has an absolute before valid time and an absolute no longer valid time so that it defines a real-time range. Relative time also has a before and after time range. However, this range is relative to some absolute real-time starting point. The relative period start point is interpreted based on the condition type that this applies to.

complexType ValidityPeriodType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

extension of `pdrl:ConditionAbstractType`

children

`ValidityPeriodAbsolute ValidityPeriodRelative`

used by

`elements PolicyEntryValidityPeriod PolicyValidityPeriod`

attributes

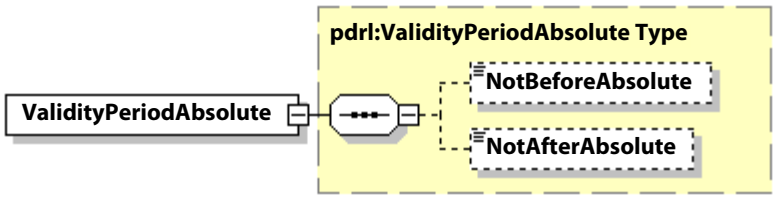
Name	Type	Use
`isAbsoluteTime`	`boolean`	required

source

```
<complexType name="ValidityPeriodType">
  <annotation>
    <documentation>Type Definition: The time period between which
an
    object is valid. It consists of the time not before and not
after.
    It can be absolute or relative. If relative, it is relative to
the
    License creation time.</documentation>
  </annotation>
  <complexContent>
    <extension base="pdrl:ConditionAbstractType">
      <choice>
        <element name="ValidityPeriodAbsolute"
type="pdrl:ValidityPeriodAbsoluteType"/>
        <element name="ValidityPeriodRelative"
type="pdrl:ValidityPeriodRelativeType"/>
      </choice>
      <attribute name="isAbsoluteTime" type="boolean"
use="required"/>
    </extension>
  </complexContent>
</complexType>
```

element ValidityPeriodType/ValidityPeriodAbsolute

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

pdrl:ValidityPeriodAbsoluteType

children

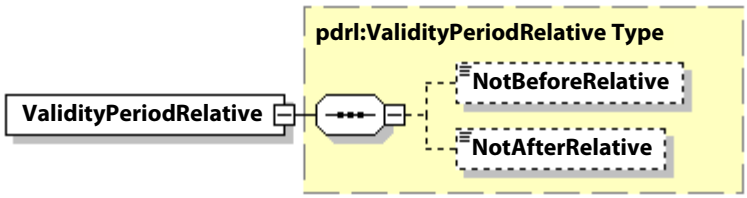
NotBeforeAbsolute NotAfterAbsolute

source

```
<element name="ValidityPeriodAbsolute"
type="pdrl:ValidityPeriodAbsoluteType"/>
```

element ValidityPeriodType/ValidityPeriodRelative

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

```
pdrl:ValidityPeriodRelativeType
```

children

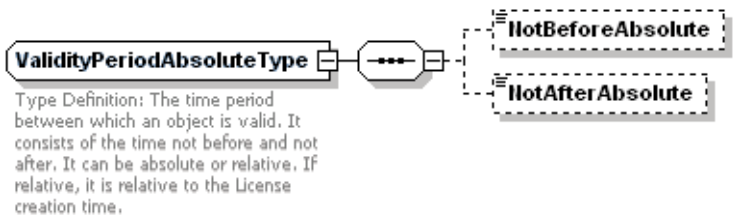
```
NotBeforeRelative NotAfterRelative
```

source

```
<element name="ValidityPeriodRelative"
type="pdrl:ValidityPeriodRelativeType"/>
```

complexType ValidityPeriodAbsoluteType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

children

NotBeforeAbsolute NotAfterAbsolute

used by

element ValidityPeriodType/ValidityPeriodAbsolute

source

```
<complexType name="ValidityPeriodAbsoluteType">
   <annotation>
      <documentation>Type Definition: The time period between which
an
      object is valid. It consists of the time not before and not
after. It
      can be absolute or relative. If relative, it is relative to the
License
      creation time.</documentation>
   </annotation>
   <sequence>
      <element name="NotBeforeAbsolute" type="dateTime"
minOccurs="0"/>
      <element name="NotAfterAbsolute" type="dateTime"
minOccurs="0"/>
   </sequence>
</complexType>
```

element ValidityPeriodAbsoluteType/NotBeforeAbsolute

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

dateTime

source

```
<element name="NotBeforeAbsolute" type="dateTime" minOccurs="0"/>
```

element ValidityPeriodAbsoluteType/NotAfterAbsolute

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

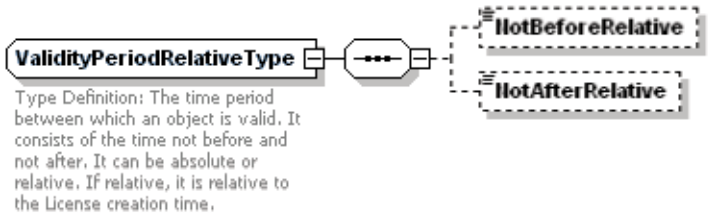
```
dateTime
```

source

```
<element name="NotAfterAbsolute" type="dateTime" minOccurs="0"/>
```

complexType ValidityPeriodRelativeType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

children

```
NotBeforeRelative NotAfterRelative
```

used by

```
element ValidityPeriodType/ValidityPeriodRelative
```

source

```
<complexType name="ValidityPeriodRelativeType">
  <annotation>
    <documentation>Type Definition: The time period between which
an
    object is valid. It consists of the time not before and not
after. It
    can be absolute or relative. If relative, it is relative to the
License
    creation time.</documentation>
  </annotation>
  <sequence>
    <element name="NotBeforeRelative" type="duration"
minOccurs="0"/>
    <element name="NotAfterRelative" type="duration"
minOccurs="0"/>
  </sequence>
</complexType>
```

element ValidityPeriodRelativeType/NotBeforeRelative

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

duration

source

```
<element name="NotBeforeRelative" type="duration" minOccurs="0"/>
```

element ValidityPeriodRelativeType/NotAfterRelative

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

duration

source

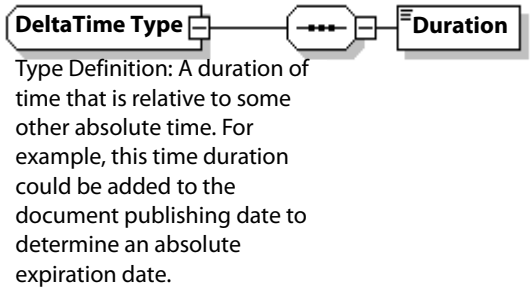
```
<element name="NotAfterRelative" type="duration" minOccurs="0"/>
```

DeltaTimeType condition

DeltaTimeType is a complex type used in the definition of time-related conditions. It represents a duration of time that is relative to some other absolute time. For example, this time duration could be added to the document publishing date to determine an absolute expiration date.

complexType DeltaTimeType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

extension of `pdrl:ConditionAbstractType`

children

`Duration`

used by

`element OfflineLeasePeriod`

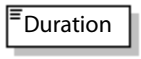
source

```
<complexType name="DeltaTimeType">
  <annotation>
    <documentation>Type Definition: A duration of time that is
relative
    to some other absolute time. For example, this time duration
could
    be added to the document publishing date to determine an
absolute
```

```
      expiration date.</documentation>
    </annotation>
    <complexContent>
      <extension base="pdrl:ConditionAbstractType">
        <sequence>
          <element name="Duration" type="duration"/>
        </sequence>
      </extension>
    </complexContent>
</complexType>
```

element DeltaTimeType/Duration

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

`duration`

source

```
<element name="Duration" type="duration"/>
```

AuditSettings condition

AuditSettings is a policy condition that defines whether a document should be tracked. If the *AuditSettings* condition is not present within the policy, the document secured by that policy will not be tracked.

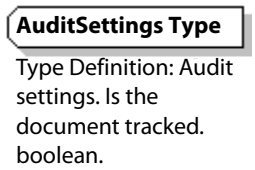
element AuditSettings

source

```
<element name="AuditSettings" type="pdrl:AuditSettingsType"
substitutionGroup="pdrl:PolicyConditionSingle">
  <annotation>
    <documentation>  Policy Condition: Audit settings. Is the
document
    tracked. boolean.</documentation>
  </annotation>
</element>
```

complexType AuditSettingsType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

extension of `pdrl:ConditionAbstractType`

used by

`element AuditSettings`

attributes

Name	Type	Use
isTracked	boolean	required

source

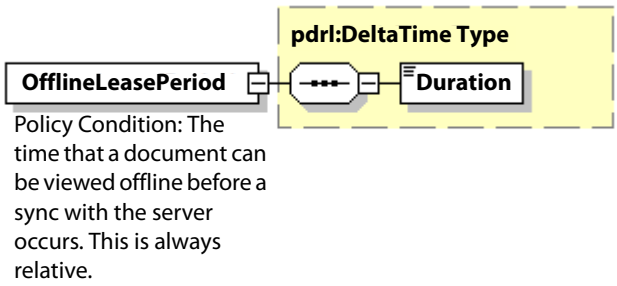
```
<complexType name="AuditSettingsType">
   <annotation>
      <documentation> Type Definition: Audit settings. Is the
document
      tracked. boolean.</documentation>
   </annotation>
   <complexContent>
      <extension base="pdrl:ConditionAbstractType">
         <attribute name="isTracked" type="boolean" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

OfflineLeasePeriod condition

OfflineLeasePeriod is the time that a document can be viewed offline before a sync with the server is required to renew the lease. This condition is always relative to the last time the client synced with the server.

element OfflineLeasePeriod

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

```
pdrl:DeltaTimeType
```

children

```
Duration
```

source

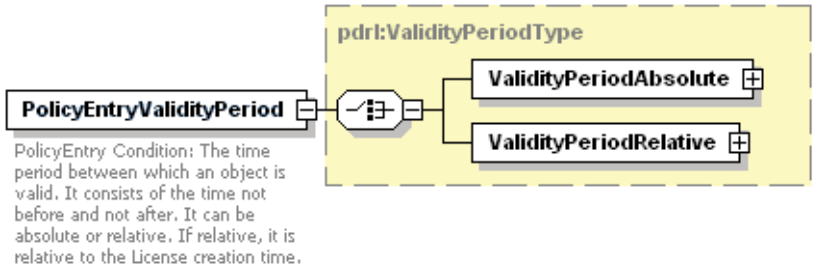
```
<element name="OfflineLeasePeriod" type="pdrl:DeltaTimeType"
substitutionGroup="pdrl:PolicyConditionSingle">
   <annotation>
      <documentation> Policy Condition: The time that a document can
be
      viewed offline before a sync with the server occurs. This is
always
      relative.</documentation>
   </annotation>
</element>
```

PolicyEntryValidityPeriod condition

PolicyEntryValidityPeriod is the time period a policy entry is valid. It consists of the time not before and not after. It can be absolute or relative. If relative, it is relative to the license creation time. This condition is typically used to allow certain users to have rights to a document for only a bounded amount of time, and other users can have permanent access. See Policy evaluation for details.

element PolicyEntryValidityPeriod

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

`pdrl:ValidityPeriodType`

children

`ValidityPeriodAbsolute ValidityPeriodRelative`

attributes

Name	Type	Use
`isAbsoluteTime`	`boolean`	required

source

```
<element name="PolicyEntryValidityPeriod"
type="pdrl:ValidityPeriodType"
substitutionGroup="pdrl:PolicyEntryConditionMulti">
   <annotation>
```

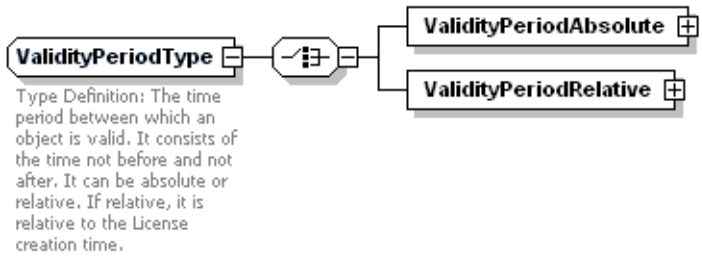
```
      <documentation> PolicyEntry Condition: The time period between
which
      an object is valid. It consists of the time not before and not
after. It
      can be absolute or relative. If relative, it is relative to the
License
      creation time.</documentation>
   </annotation>
</element>
```

PolicyValidityPeriod condition

PolicyValidityPeriod is the time period a policy is valid. It consists of the time not before and not after. It can be absolute or relative. If relative, it is relative to the license creation time. This condition is typically used to allow access to a resource for only a bounded amount of time. See Policy evaluation for details.

element PolicyValidityPeriod

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl

type

`pdrl:ValidityPeriodType`

children

`ValidityPeriodAbsolute ValidityPeriodRelative`

attributes

Name	Type	Use
isAbsoluteTime	boolean	required

source

```
<element name="PolicyValidityPeriod"
type="pdrl:ValidityPeriodType"
substitutionGroup="pdrl:PolicyConditionSingle">
   <annotation>
      <documentation> Policy Condition: The time period between
which an
```

```
     object is valid. It consists of the time not before and not
after. It
     can be absolute or relative. If relative, it is relative to the
License
     creation time.</documentation>
   </annotation>
</element>
```

Watermark condition

Watermark defines whether a watermark is needed and, if so, what template to use. The watermark is identified in the *TemplateID* element using the watermark's unique ID. If this condition is not present, no watermark is applied to the document.

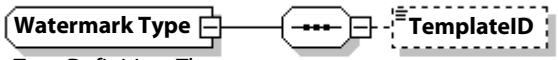
element Watermark

source

```
<element name="Watermark" type="pdrl:WatermarkType"
substitutionGroup="pdrl:PolicyConditionSingle">
  <annotation>
    <documentation> Policy Condition: Defines whether a watermark
is
    needed and if so what template to use.</documentation>
  </annotation>
</element>
```

complexType WatermarkType

diagram



Type Definition: The type of watermarking required for the document.

namespace

http://www.adobe.com/schema/1.0/pdrl

type

extension of `pdrl:ConditionAbstractType`

children

`TemplateID`

used by

`element Watermark`

attributes

Name	Type	Use
isWatermarked	boolean	required

source

```
<complexType name="WatermarkType">
   <annotation>
      <documentation> Type Definition: The type of watermarking
required
      for the document.</documentation>
   </annotation>
   <complexContent>
      <extension base="pdrl:ConditionAbstractType">
         <sequence>
            <element name="TemplateID" type="string" minOccurs="0"/>
         </sequence>
         <attribute name="isWatermarked" type="boolean"
use="required"/>
      </extension>
   </complexContent>
</complexType>
```

element WatermarkType/TemplateID

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl

type

string

source

```
<element name="TemplateID" type="string" minOccurs="0"/>
```

5 Extension Schema

This chapter describes conditions and permissions that are specific to Acrobat and LiveCycle Rights Management ES3. See Core Schema and Extensibility for the complete schema definition.

Acrobat conditions

This condition element defines Acrobat-specific policy conditions. It is extended from *PolicyConditionSingle*, which means that it is a policy-level condition.

Elements:

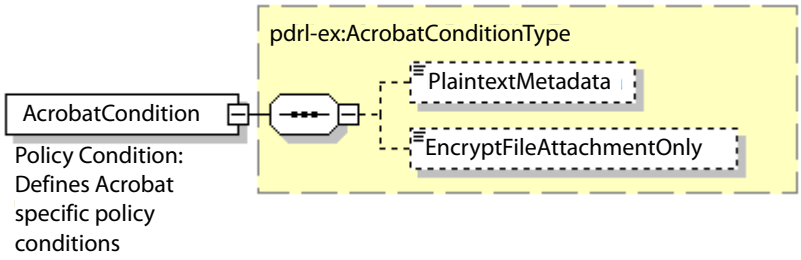
`PlaintextMetadata`: Boolean type. If `TRUE`, metadata is stored as clear-text in the document. If `FALSE` or if not defined, metadata is stored encrypted in the document.

`EncryptFileAttachmentOnly`: Boolean type. If the value is `TRUE`, the primary document is in clear text, and only file attachments are encrypted. If the value is `FALSE` or if not defined, the entire document is encrypted.

Note: If `EncryptFileAttachmentOnly` is `TRUE`, then metadata is always be stored in clear text regardless of what the `PlaintextMetadata` flag is set to. If `EncryptFileAttachmentOnly` is `FALSE`, then the value of `PlaintextMetadata` determines how the metadata gets stored.

element AcrobatCondition

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl-ex

type

`pdrl-ex:AcrobatConditionType`

children

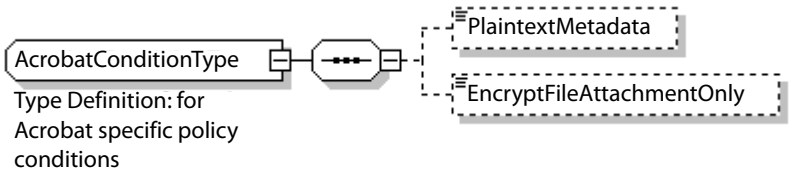
`PlaintextMetadata EncryptFileAttachmentOnly`

source

```
   <element name="AcrobatCondition"
type="pdrl-ex:AcrobatConditionType"
substitutionGroup="pdrl:PolicyConditionSingle">
     <annotation>
       <documentation> Policy Condition: Defines Acrobat specific
policy
       conditions</documentation>
     </annotation>
   </element>
```

complexType AcrobatConditionType

diagram



namespace

http://www.adobe.com/schema/1.0/pdrl-ex

type

extension of `pdrl:ConditionAbstractType`

children

`PlaintextMetadata EncryptFileAttachmentOnly`

used by

`element AcrobatCondition`

source

```
   <complexType name="AcrobatConditionType">
     <annotation>
       <documentation> Type Definition: for Acrobat specific policy
       conditions.</documentation>
     </annotation>
     <complexContent>
       <extension base="pdrl:ConditionAbstractType">
         <sequence>
```

```
            <element name="PlaintextMetadata" type="boolean"
default="false" minOccurs="0"/>
            <element name="EncryptFileAttachmentOnly"
type="boolean" default="false" minOccurs="0"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
```

element AcrobatConditionType/PlaintextMetadata

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl-ex

type

boolean

source

```
  <element name="PlaintextMetadata" type="boolean" default="false"
minOccurs="0"/>
```

element AcrobatConditionType/EncryptFileAttachmentOnly

diagram

namespace

http://www.adobe.com/schema/1.0/pdrl-ex

type

boolean

source

```
<element name="EncryptFileAttachmentOnly" type="boolean"
default="false" minOccurs="0"/>
```

Acrobat and Rights Management ES3 permissions

This list of permissions are enforced by Acrobat and Rights Management ES3. The definition of these permissions and how they map to Acrobat permissions are found in the Acrobat Permissions Engineering Specification [A-PERM].

simpleType PermissionNameEnum

namespace

http://www.adobe.com/schema/1.0/pdrl-ex

type

restriction of `QName`

facets

Facet	Description
`enumeration pdrl-ex:com.adobe.aps.onlineOpen`	Allow user to open document online.
`enumeration pdrl-ex:com.adobe.aps.offlineOpen`	Allow user to open document online and offline.
`enumeration pdrl-ex:com.adobe.aps.revoke`	Allow user the ability to revoke document access privileges.
`enumeration pdrl-ex:com.adobe.aps.policySwitch`	Allow user the ability to switch policy privileges.
`enumeration pdrl-ex:com.adobe.aps.pdf.printHigh`	Allow user to print with high and low resolution.
`enumeration pdrl-ex:com.adobe.aps.pdf.printLow`	Allow user to print with low resolution only.
`enumeration pdrl-ex:com.adobe.aps.pdf.edit`	Allow user to edit the document.
`enumeration pdrl-ex:com.adobe.aps.pdf.docAssembly`	Allow user to insert, delete, and rotate pages.
`enumeration pdrl-ex:com.adobe.aps.pdf.editNotes`	Allow user to use the Acrobat commenting tools.
`enumeration pdrl-ex:com.adobe.aps.pdf.fillAndSign`	Allow user to fill in form fields and digitally sign the document.

Facet	Description
`enumeration pdrl-ex:com.adobe.aps.pdf.copy`	Allow user to have replication capabilities, including copying of text, images, and other content.
`enumeration pdrl-ex:com.adobe.aps.pdf.accessible`	Allow user to use the document with a screen reader.

6 Security Considerations

Policies and *licenses* are used to convey rights to protected resources. Therefore, the security of persisting and transmitting these objects in a distributed environment is essential to the overall security of a rights management system.

Integrity

Licenses and policies are stored persistently within the content of a document. An application that tries to enforce rights based on these objects needs to validate their integrity. Without some form of integrity protection, malicious users could change their permissions in the policy (for example) and grant rights that were not intended.

The license object contains the follow optional elements:

```
<element name="HMAC" type="pdrl:HMACType" minOccurs="0"/>
<element ref="ds:Signature" minOccurs="0"/>
```

The issuing authority should protect the integrity of the license by either including an HMAC of the entire license or applying an XML digital signature to the license as defined in [DSIG]. The client can then verify the integrity of the license by validating the signature or HMAC.

Privacy

The policy and license contain potentially sensitive information about the resource being protected. They contain information covering who is allowed to access a document, when the document expires, who published the document, etc. This information should be encrypted when persisted within a document file or while on disk so that malicious users cannot use it to plan attacks.

It is recommended that all implementations using PDRL encrypt this information whenever it is persisted.

It is also important that the data be protected while transferring these objects between a client and server in a distributed system. It is recommended that the implementation use SSL or some other transport protocol encryption algorithm for ensuring the privacy of this data while in transit.

Trust model

A trust model needs to be implemented to ensure that licenses and policies are managed and issued from a trusted source. The digital signature or HMAC of the license can be used for this purpose. The client should be configured out of band to trust licenses that are issued by the issuing authority.

In the case of digital signatures, the signing certificate should belong to the issuing authority. The client should be pre-configured with the issuing authority's public certificate. When the client validates the digital signature on the license, it should also verify that the signing certificate matches the pre-configured issuing authority's certificate, which it already trusts as a valid signer.

The case of HMAC is similar. The client should obtain a shared secret key from the issuing authority out of band. The client then uses that shared secret key to verify that the license was issued by a trusted source.

The specific implementation details of the trust model are outside the scope of this document.

Authentication

Before transferring licenses and policies between the client and server, a mutual authentication should occur. This will allow the client to know that it is receiving data from a trusted source. It will also allow the server to know that it is not giving data to an client that is not trusted.

The details of the authentication mechanism and the transport protocol are implementation-specific and outside of the scope of this document.

7 References

The following documents are referenced in this document.

Document	Description
PDF	*PDF Reference, sixth edition, Adobe Portable Document Format, Version 1.7*, Adobe Systems Incorporated
DSIG	D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web 2051 Consortium, February 2002. http://www.w3.org/TR/xmldsig-core/

Copyright notices

Adobe® LiveCycle® ES3 (10.0.2) Portable Document Rights Language (PDRL) Help for Microsoft® Windows®, Linux®, and Solaris®
Edition 2.0, April 2012

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.