



**Adobe**

# **Portable Document Rights Language (PDRL) Specification**

July 2006

Version 7.2

© 2006 Adobe Systems Incorporated. All rights reserved.

Adobe® LiveCycle™ Policy Server 7.2 Portable Document Rights Language (PDRL) Specification  
July 2006

If this guide is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe Systems Incorporated. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end user license agreement.

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Please remember that existing artwork or images that you may want to include in your project may be protected under copyright law. The unauthorized incorporation of such material into your new work could be a violation of the rights of the copyright owner. Please be sure to obtain any permission required from the copyright owner.

Any references to company names and company logos in sample material or in the sample forms included in this software are for demonstration purposes only and are not intended to refer to any actual organization.

Adobe, the Adobe logo, Acrobat, and LiveCycle are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

All other trademarks are the property of their respective owners.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org>).

This product contains either BSAFE and/or TIPEM Software by RSA Data Security, Inc.

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

Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

# Contents

<b>1</b>	<b>Introduction .....</b>	<b>6</b>
	About Adobe Portable Document Rights Language .....	6
	Requirements.....	6
	Design Goals and Principles .....	7
	Non-Goals.....	7
<b>2</b>	<b>Concepts .....</b>	<b>8</b>
	Data Model .....	8
	License.....	9
	Resource .....	10
	Policy.....	10
	Policy Evaluation .....	13
<b>3</b>	<b>Extensibility .....</b>	<b>15</b>
	Policy Extensibility .....	15
	Permissions .....	15
	Conditions .....	16
	Properties.....	16
	License Extensibility .....	17
<b>4</b>	<b>Core Schema.....</b>	<b>18</b>
	License Element.....	18
	element License .....	19
	complexType LicenseType.....	19
	IssuingAuthority Element .....	20
	element LicenseType/IssuingAuthority .....	20
	PolicyIDReference Element .....	20
	element PolicyIDReference .....	21
	complexType PolicyIDReferenceType .....	21
	Property Element .....	21
	element Property.....	21
	complexType PropertyType.....	22
	element PropertyType/PropertyValue.....	22
	HMAC Element.....	22
	element LicenseType/HMAC .....	22
	simpleType HMACType .....	23
	ds:Signature Element.....	23
	Policy Element .....	23
	element Policy .....	23
	complexType PolicyType .....	24
	PolicyEntry Element .....	25
	element PolicyEntry.....	25
	complexType PolicyEntryType .....	26

**4 Core Schema (Continued)**

Principal Element ..... 27

    element Principal ..... 27

    complexType PrincipalType ..... 27

    element PrincipalType/PrincipalDomain ..... 27

    element PrincipalType/PrincipalName ..... 28

    simpleType PrincipalNameTypeEnumeration ..... 28

Permission Element ..... 28

    element Permission ..... 29

    complexType PermissionType ..... 29

    simpleType PermissionAccessType ..... 29

PolicyEntryCondition Elements ..... 29

Property Element ..... 30

Resource Element ..... 30

    element Resource ..... 30

    complexType ResourceType ..... 30

Publisher Element ..... 31

    element ResourceType/Publisher ..... 31

PublishTime Element ..... 31

    element ResourceType/PublishTime ..... 31

ResourceName Element ..... 31

    element ResourceType/ResourceName ..... 31

ResourceLocation Element ..... 32

    element ResourceType/ResourceLocation ..... 32

ResourceID Element ..... 32

    element ResourceType/ResourceID ..... 32

Condition Elements ..... 32

    ConditionAbstractType type ..... 32

    complexType ConditionAbstractType ..... 32

PolicyCondition and PolicyConditionEntry Base Elements ..... 33

    element PolicyConditionMulti ..... 33

    element PolicyConditionSingle ..... 34

    element PolicyEntryConditionMulti ..... 35

    element PolicyEntryConditionSingle ..... 35

ValidityPeriodType Condition ..... 36

    complexType ValidityPeriodType ..... 36

    element ValidityPeriodType/ValidityPeriodAbsolute ..... 37

    element ValidityPeriodType/ValidityPeriodRelative ..... 37

    complexType ValidityPeriodAbsoluteType ..... 37

    element ValidityPeriodAbsoluteType/NotBeforeAbsolute ..... 38

    element ValidityPeriodAbsoluteType/NotAfterAbsolute ..... 38

    complexType ValidityPeriodRelativeType ..... 38

    element ValidityPeriodRelativeType/NotBeforeRelative ..... 39

    element ValidityPeriodRelativeType/NotAfterRelative ..... 39

DeltaTimeType Condition ..... 39

    complexType DeltaTimeType ..... 39

    element DeltaTimeType/Duration ..... 40

---

**4 Core Schema (Continued)**

- AuditSettings Condition ..... 40
  - element AuditSettings ..... 40
  - complexType AuditSettingsType ..... 41
- OfflineLeasePeriod Condition ..... 41
  - element OfflineLeasePeriod ..... 41
- PolicyEntryValidityPeriod Condition ..... 42
  - element PolicyEntryValidityPeriod ..... 42
- PolicyValidityPeriod Condition ..... 42
  - element PolicyValidityPeriod ..... 43
- Watermark Condition ..... 43
  - element Watermark ..... 43
  - complexType WatermarkType ..... 44
  - element WatermarkType/TemplateID ..... 44

**5 Extension Schema for LiveCycle Policy Server and Acrobat ..... 45**

- Acrobat Conditions ..... 45
  - element AcrobatCondition ..... 45
  - complexType AcrobatConditionType ..... 46
  - element AcrobatConditionType/PlaintextMetadata ..... 46
  - element AcrobatConditionType/EncryptFileAttachmentOnly ..... 46
- Acrobat and LiveCycle Policy Server Permissions ..... 47
  - simpleType PermissionNameEnum ..... 47

**6 Security Considerations ..... 48**

- Integrity ..... 48
- Privacy ..... 48
- Trust Model ..... 49
- Authentication ..... 49

**7 References ..... 50**

## About Adobe Portable Document Rights Language

Portable Document Rights Language (PDRL) is a language for expressing rights and conditions for accessing digital content. This language was created as part of the Adobe® LiveCycle™ Policy Server development effort. It was originally devised as an enhancement to the permission language that has existed in Adobe Acrobat® Professional and Acrobat Standard to secure Adobe PDF. To support a rich set of document control features, such as expiration, revocation, and access control lists (ACLs), the design team realized the need to create a language for the expression of these new features. This idea evolved into the concept of *policies*. An end-user can bundle together a set of rights and conditions into a container known as a *policy*. Then the policy can be applied to new documents that need similar protections over time. This powerful concept allows business security rules to be created and reused in an intelligent document processing workflow.

As part of the investigation into the development of PDRL, the team evaluated the possibility of using an existing standard rights language instead of developing one for our purposes. The investigation led to the conclusion that this is still an emerging space without a clearly accepted standard. Therefore, without a clear standard on which to base this work, the team chose to create its own rights language to ensure that it could meet all the feature requirements and allow for a faster time to market. Industry-standard XML and XML Schema were chosen as the syntax to express PDRL because of their pervasiveness and wide acceptance.

While PDRL was first devised to meet the needs of the LiveCycle Policy Server, it was also important to be able to evolve the language over time; therefore, language features were added to accommodate this requirement. PDRL was also designed in an abstract enough way so that it could potentially be used in other implementations unrelated to document access control.

## Requirements

PDRL meets the following requirements:

**Document control needs:** PDRL allows for the expression of the document control features implemented in the LiveCycle Policy Server product.

**Secure:** The language provides features that allow it to be used for securely transmitting and attaching rights to protected resources.

**Extensible:** As new document control features are requested over time, the language is extensible enough to handle them.

**PDF compatible:** The language is designed to be compatible with the PDF public specification [PDF]. Policies and licenses described within this document can be contained within a PDF file.

## Design Goals and Principles

The following goals were taken into account as part of the development of PDRL.

**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 LiveCycle Policy Server, it is abstract enough for use in expressing other types of policies in other implementations. For example, a *resource* is a very generic concept. LiveCycle Policy Server 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 LiveCycle Policy Server 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.

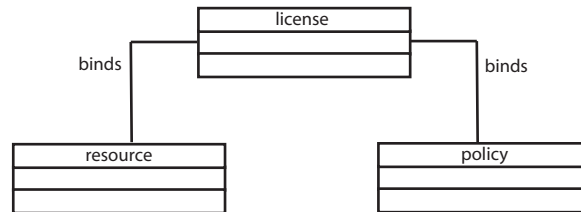
## Non-Goals

Standardization is not addressed in this document. This document was not written with the intent of submission to a standards body. If that is preferred in the future, the document would need to be reformatted with standardization in mind. PDRL itself was written in a generic and abstract manner; therefore, it is conceivable that the language could be submitted to a standards body for use by other companies in their own rights management implementations independent of PDF and LiveCycle Policy Server.

## 2

## 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

To best describe how these elements work together, let's use LiveCycle Policy Server and its integration with Acrobat 7.0 as an example.

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

The first thing a user does is create a policy. He logs into LiveCycle Policy Server and creates a policy that contains 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 LiveCycle Policy Server database.

The next step is for a user to open a PDF document using Acrobat and attempt to apply security to that document. The Acrobat UI 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 LiveCycle Policy Server. LiveCycle Policy Server 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 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 "live" independently from one another and can be converted to/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 example 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-CCFFFA2087E</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. For LiveCycle Policy Server, a *resource* is the document being controlled. The term *resource* is purposefully abstract. The intent is for a resource to refer to other content types over time. Therefore, policies can be used in other scenarios outside the scope of just document control.

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 example 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-CCFFFA2087E </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 does not get 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 scenario. For example, an administrator may create a policy named “Company Confidential” that ensures that corporate documents are only accessible to company employees. Then, 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, it was a goal to provide a basic set of policy building blocks within the core schema and then allow future users of this specification to extend 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 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 example shows a typical policy.

```

1. <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2. <Policy PolicyID="4F3F323D-5C45-3031-8A52-F6151AB82927" PolicyInstanceVersion="1"
3. xmlns="http://www.adobe.com/schema/1.0/pdrl" xmlns:pdrl-ex="http://www.adobe.com/schema/1.0/pdrl-ex"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:schemaLocation="http://www.adobe.com/schema/1.0/pdrl-ex ..\pdrl-ex.xsd">
6.   <PolicyEntry>
7.     <Permission PermissionName="pdrl-ex:com.adobe.aps.onlineOpen" Access="ALLOW"/>
8.     <Permission PermissionName="pdrl-ex:com.adobe.aps.offlineOpen" Access="ALLOW"/>
9.     <Permission PermissionName="pdrl-ex:com.adobe.aps.policySwitch" Access="ALLOW"/>
10.    <Permission PermissionName="pdrl-ex:com.adobe.aps.revoke" Access="ALLOW"/>
11.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.printHigh" Access="ALLOW"/>
12.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.printLow" Access="ALLOW"/>
13.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.copy" Access="ALLOW"/>
14.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.edit" Access="ALLOW"/>
15.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.editNotes" Access="ALLOW"/>
16.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.fillAndSign" Access="ALLOW"/>
17.    <Principal PrincipalNameType="SYSTEM">
18.      <PrincipalDomain>EDC_SPECIAL</PrincipalDomain>
19.      <PrincipalName>publisher</PrincipalName>
20.    </Principal>
21.  </PolicyEntry>
22.  <PolicyEntry>
23.    <Permission PermissionName="pdrl-ex:com.adobe.aps.onlineOpen" Access="ALLOW"/>
24.    <Permission PermissionName="pdrl-ex:com.adobe.aps.offlineOpen" Access="ALLOW"/>
25.    <Permission PermissionName="pdrl-ex:com.adobe.aps.policySwitch" Access="ALLOW"/>
26.    <Permission PermissionName="pdrl-ex:com.adobe.aps.revoke" Access="ALLOW"/>
27.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.printHigh" Access="ALLOW"/>
28.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.printLow" Access="ALLOW"/>
29.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.copy" Access="ALLOW"/>
30.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.edit" Access="ALLOW"/>
31.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.editNotes" Access="ALLOW"/>
32.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.fillAndSign" Access="ALLOW"/>
33.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.accessible" Access="ALLOW"/>
34.    <PolicyEntryValidityPeriod isAbsoluteTime="true">
35.      <ValidityPeriodAbsolute>
36.        <NotBeforeAbsolute>2004-06-04T10:00:00+00:00</NotBeforeAbsolute>
37.        <NotAfterAbsolute>2004-07-05T10:00:00+00:00</NotAfterAbsolute>
38.      </ValidityPeriodAbsolute>
39.    </PolicyEntryValidityPeriod>
40.    <Principal PrincipalNameType="USER">
41.      <PrincipalDomain>adobe.com</PrincipalDomain>
42.      <PrincipalName>uid=jsonfili,ou=people,o=adobe.com</PrincipalName>
43.    </Principal>
44.  </PolicyEntry>
45.  <PolicyEntry>
46.    <Permission PermissionName="pdrl-ex:com.adobe.aps.onlineOpen" Access="ALLOW"/>
47.    <Permission PermissionName="pdrl-ex:com.adobe.aps.offlineOpen" Access="ALLOW"/>
48.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.accessible" Access="ALLOW"/>
49.    <Permission PermissionName="pdrl-ex:com.adobe.aps.pdf.printHigh" Access="ALLOW"/>

```

```

50.     <Principal PrincipalNameType="USER">
51.         <PrincipalDomain>adobe.com</PrincipalDomain>
52.         <PrincipalName>uid=jpravetz,ou=people,o=adobe.com</PrincipalName>
53.     </Principal>
54.     <Principal PrincipalNameType="USER">
55.         <PrincipalDomain>adobe.com</PrincipalDomain>
56.         <PrincipalName>uid=jsanfili,ou=people,o=adobe.com</PrincipalName>
57.     </Principal>
58. </PolicyEntry>
59. <PolicyEntry>
60.     <Permission PermissionName="pdr-ex:com.adobe.aps.onlineOpen" Access="ALLOW"/>
61.     <Permission PermissionName="pdr-ex:com.adobe.aps.offlineOpen" Access="ALLOW"/>
62.     <Permission PermissionName="pdr-ex:com.adobe.aps.pdf.accessible" Access="ALLOW"/>
63.     <Permission PermissionName="pdr-ex:com.adobe.aps.pdf.printLow" Access="ALLOW"/>
64.     <Permission PermissionName="pdr-ex:com.adobe.aps.pdf.fillAndSign" Access="ALLOW"/>
65.     <Principal PrincipalNameType="GROUP">
66.         <PrincipalDomain>adobe.com</PrincipalDomain>
67.         <PrincipalName>cn=jsanfili-direct reports,ou=groups,o=adobe.com</PrincipalName>
68.     </Principal>
69. </PolicyEntry>
70. <Watermark isWatermarked="true">
71.     <TemplateID>FEF70094-447F-07C5-EC13-01A6BEC4C2CC </TemplateID>
72. </Watermark>
73. <pdr-ex:AcrobatCondition>
74.     <pdr-ex:PlaintextMetadata>>true</pdr-ex:PlaintextMetadata>
75.     <pdr-ex:EncryptFileAttachmentOnly>>false</pdr-ex:EncryptFileAttachmentOnly>
76. </pdr-ex:AcrobatCondition>
77. <PolicyValidityPeriod isAbsoluteTime="false">
78.     <ValidityPeriodRelative>
79.         <NotAfterRelative>P30D</NotAfterRelative>
80.     </ValidityPeriodRelative>
81. </PolicyValidityPeriod>
82. <OfflineLeasePeriod>
83.     <Duration>P3D</Duration>
84. </OfflineLeasePeriod>
85. <Property PropertyName="DocumentumProperty1">
86.     <PropertyValue>value1</PropertyValue>
87.     <PropertyValue>value2</PropertyValue>
88. </Property>
89. </Policy>

```

Let's examine each part of this sample policy to highlight some of the key features:

- Lines 6-65: The policy contains 4 policy entries. A policy can contain 0 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 policy level validity period. See ["Policy Evaluation" on page 13](#) for the policy evaluation algorithm.

- Lines 46-58: This is an example of a policy entry that contains multiple principals. Both “jsanfilii” 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” on page 13](#) 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 LiveCycle Policy Server 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: This example defines 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: This example is of 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 Acrobat 7.0 and LiveCycle Policy Server 7.0, 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

The PDRL schema is designed to be extensible. There are several areas in which the schema can be extended. It was a design goal to provide a basic set of building blocks within the core schema and then allow future users of this specification to extend the schema to add 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 LiveCycle Policy Server.

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>
```

Then a separate schema, "pdrl-ex.xsd" contains the LiveCycle Policy Server specific permission names.

```
<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.

## 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 example shows a condition used for policy entries.

```
<element name="PolicyEntryValidityPeriod" type="pdr1:ValidityPeriodType"
substitutionGroup="pdr1: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="pdr1:ValidityPeriodType"
substitutionGroup="pdr1: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 example shows a condition that applies to the entire policy.

```
<element name="AuditSettings" type="pdr1:AuditSettingsType"
substitutionGroup="pdr1: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 (integers, strings, Booleans, ...), 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 example shows how a property can be used within a policy.

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

## License Extensibility

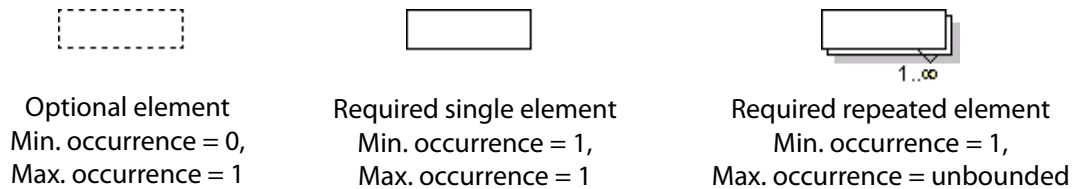
The license can be extended through properties. This type definition is the same one described in [“Properties” on page 16](#).

# 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.

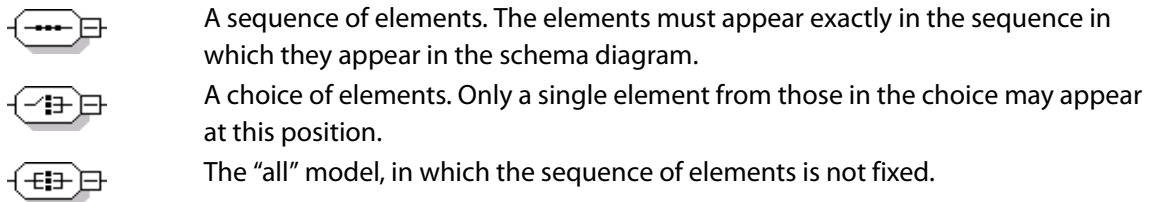
The cardinality of the element (0..1, 1 exactly, 0..n, 1..n) is indicated by the border of the elements. Optional elements are drawn with a dashed line and required elements with a solid line. A maximum occurrence greater than one is indicated by a double border.



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.



Other symbols include these.



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.

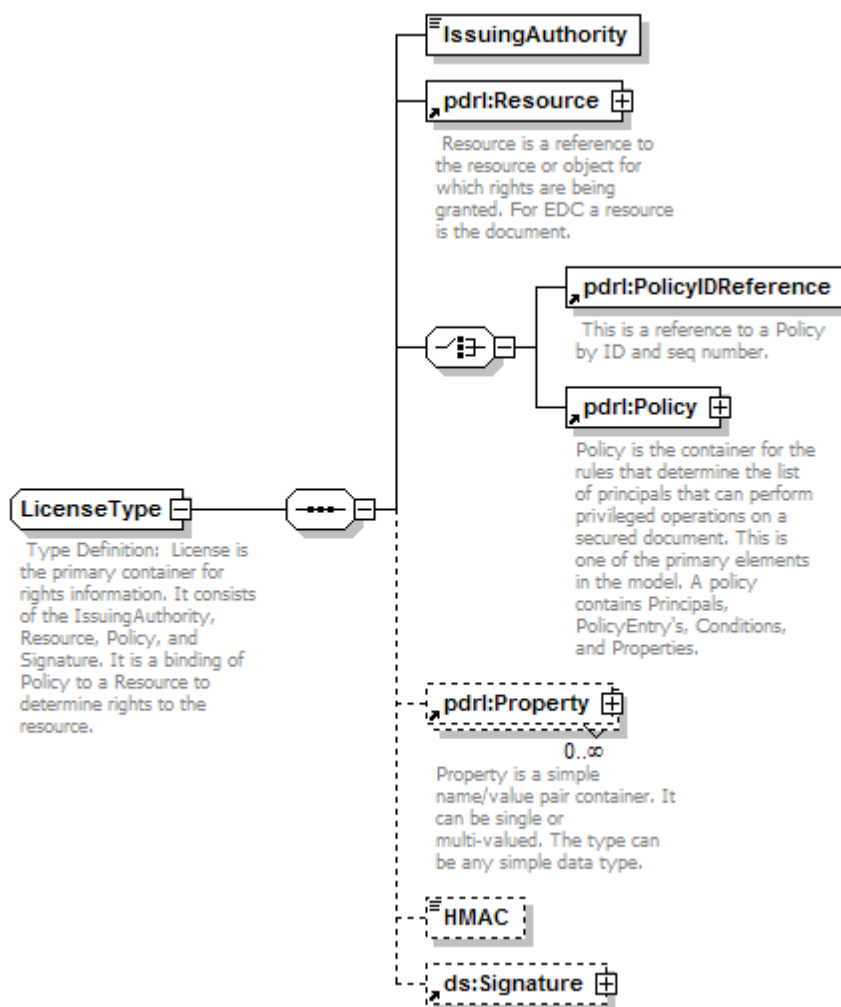
**LicenseSchemaVersion:** The version of the schema used by this instance. It is optional when the instance is created, but the server must fill in a value before it can be used in a security workflow.

## 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



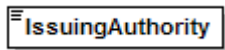
namespace	http://www.adobe.com/schema/1.0/pdrl				
children	IssuingAuthority pdrl:Resource pdrl:PolicyIDReference pdrl:Policy pdrl:Property HMAC ds:Signature				
used by	element License				
attributes	Name	Type	Use	Default	Fixed
	LicenseID	string	optional		
	LicenseInstanceVersion	int	optional		
	LicenseIssueTime	dateTime	optional		
	LicenseSchemaVersion	string	optional		
source	<pre> &lt;complexType name="LicenseType"&gt;   &lt;annotation&gt;     &lt;documentation&gt; 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.&lt;/documentation&gt;   &lt;/annotation&gt;   &lt;sequence&gt;     &lt;element name="IssuingAuthority" type="anyURI"/&gt;     &lt;element ref="pdrl:Resource"/&gt;     &lt;choice&gt;       &lt;element ref="pdrl:PolicyIDReference"/&gt;       &lt;element ref="pdrl:Policy"/&gt;     &lt;/choice&gt;     &lt;element ref="pdrl:Property" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;element name="HMAC" type="pdrl:HMACType" minOccurs="0"/&gt;     &lt;element ref="ds:Signature" minOccurs="0"/&gt;   &lt;/sequence&gt;   &lt;attribute name="LicenseID" type="string" use="optional"/&gt;   &lt;attribute name="LicenseInstanceVersion" type="int" use="optional"/&gt;   &lt;attribute name="LicenseIssueTime" type="dateTime" use="optional"/&gt;   &lt;attribute name="LicenseSchemaVersion" type="string" use="optional"/&gt; &lt;/complexType&gt; </pre>				

## 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

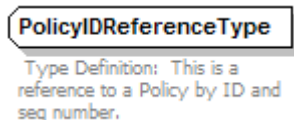
*PolicyIDReference* is a reference to a policy by `policyID`. `PolicyID` must be a unique identifier.

## 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	Default	Fixed
	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.

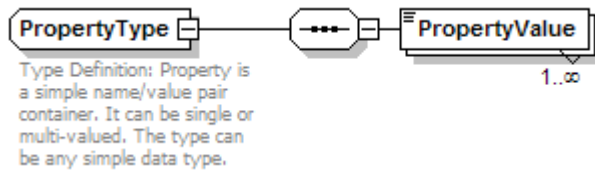
`PropertyNamespace`: Optional attribute and can be used to group "like" properties into a single category.

## 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



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

children `PropertyValue`

used by `element Property`

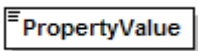
attributes	Name	Type	Use	Default	Fixed
	<code>PropertyName</code>	<code>string</code>	<code>required</code>		
	<code>PropertyNamespace</code>	<code>string</code>	<code>optional</code>		

```
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 integrity-protect 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

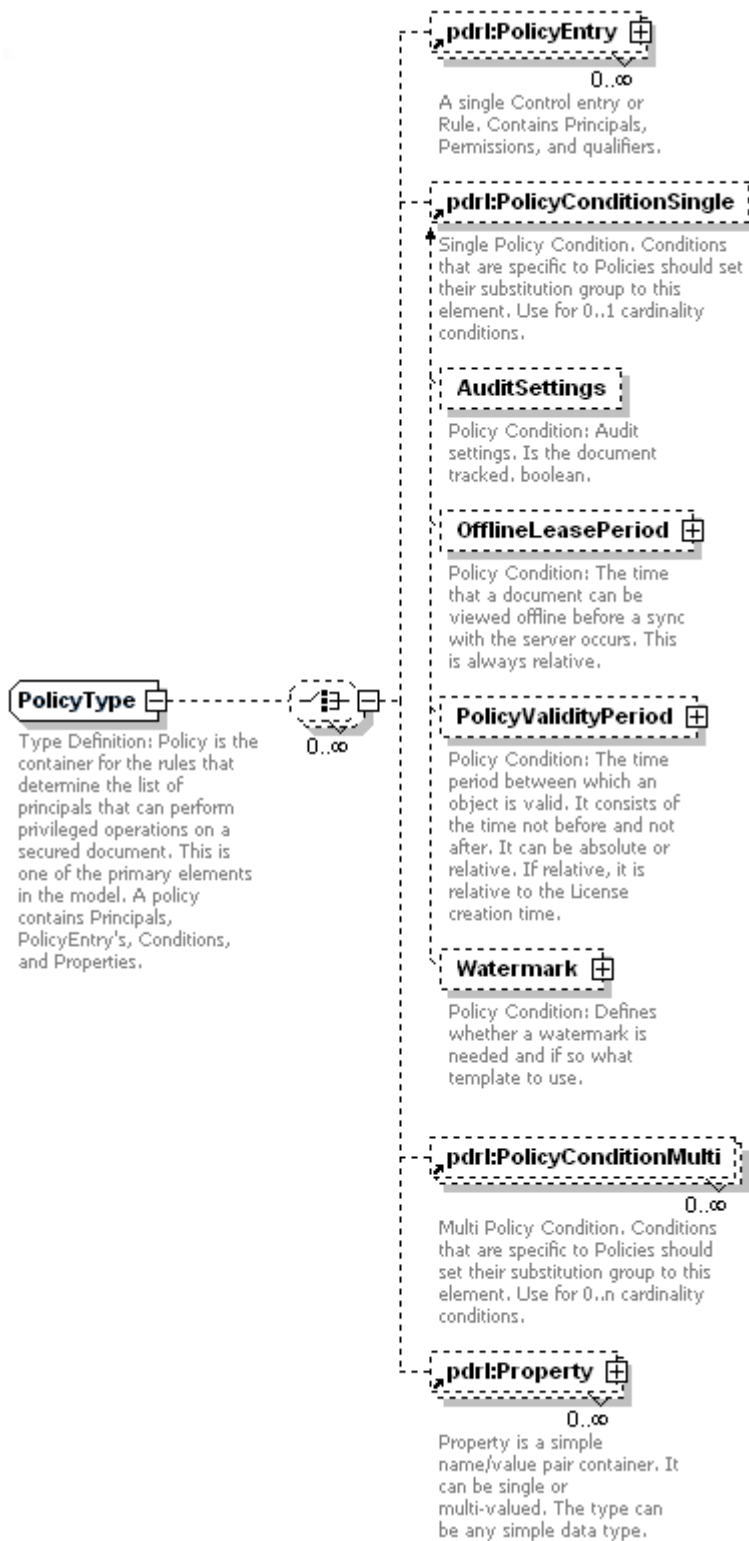
```

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



namespace <http://www.adobe.com/schema/1.0/pdrl>

children `pdr:PolicyEntry` `pdr:PolicyConditionSingle` `pdr:PolicyConditionMulti`  
`pdr:Property`

used by	element Policy				
attributes	Name	Type	Use	Default	Fixed
	PolicyID	string	optional		
	PolicyInstanceVersion	int	optional		
	PolicySchemaVersion	string	optional		
	PolicyCreationTime	dateTime	optional		
source	<pre> &lt;complexType name="PolicyType"&gt;   &lt;annotation&gt;     &lt;documentation&gt;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.&lt;/documentation&gt;     &lt;/annotation&gt;     &lt;choice minOccurs="0" maxOccurs="unbounded"&gt;       &lt;element ref="pdrl:PolicyEntry" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;element ref="pdrl:PolicyConditionSingle" minOccurs="0"/&gt;       &lt;element ref="pdrl:PolicyConditionMulti" minOccurs="0"         maxOccurs="unbounded"/&gt;       &lt;element ref="pdrl:Property" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/choice&gt;     &lt;attribute name="PolicyID" type="string" use="optional"/&gt;     &lt;attribute name="PolicyInstanceVersion" type="int" use="optional"/&gt;     &lt;attribute name="PolicySchemaVersion" type="string" use="optional"/&gt;     &lt;attribute name="PolicyCreationTime" type="dateTime" use="optional"/&gt;   &lt;/complexType&gt; </pre>				

## 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.

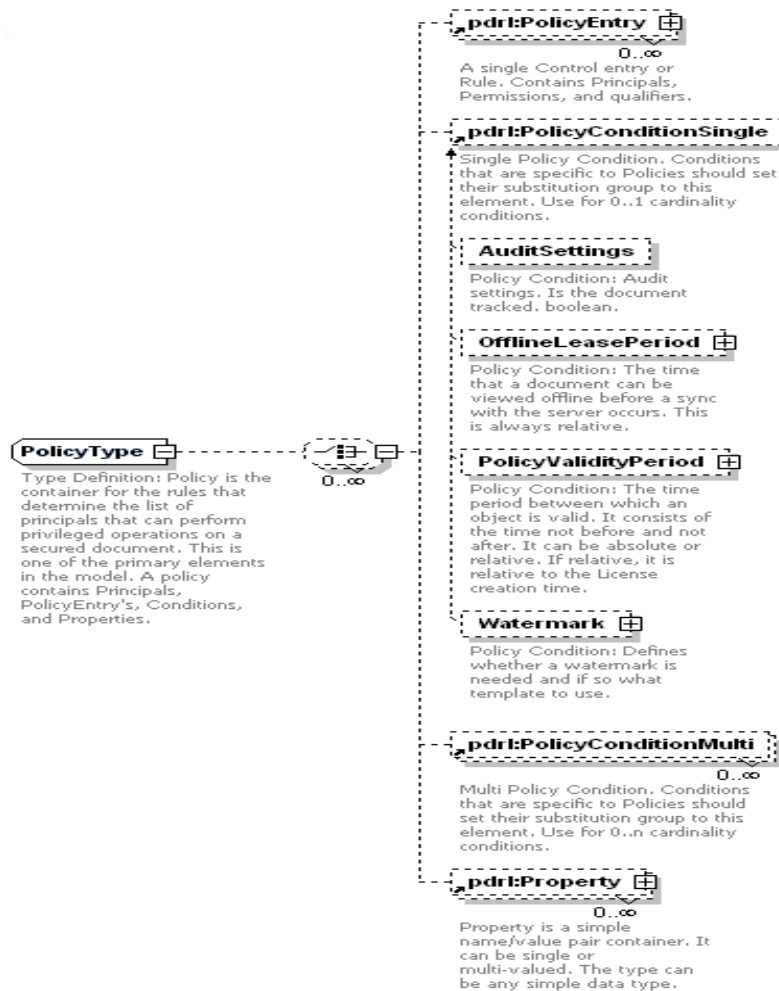
```

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



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

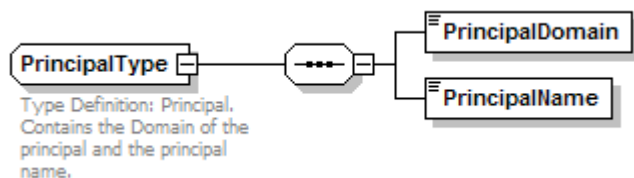
*Principal* contains the domain of the principal and the name of the principal. *PrincipalDomain* represents the user directory that stores the principal identities, and *PrincipalName* is the unique user ID within that user directory.

### 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

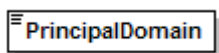
used by elements Principal ResourceType/Publisher

attributes	Name	Type	Use	Default	Fixed
	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



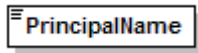
namespace <http://www.adobe.com/schema/1.0/pdrl>

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=jsanfilii,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.

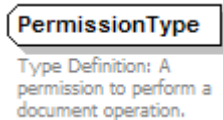
Access: Is the permission enforcement algorithm supposed to ALLOW or DENY this right to the principal.

## 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	Default	Fixed
	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

[See "Condition Elements" on page 32.](#)

## Property Element

[See "Property Element" on page 21.](#)

## Resource Element

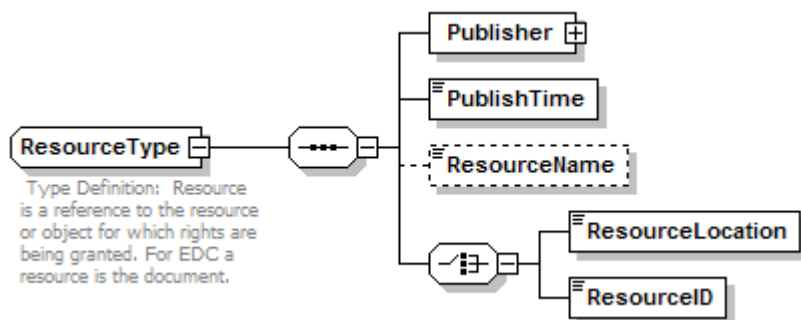
*Resource* is a reference to the resource or object for which rights are being granted. For EDC, a resource is the document. [See "Resource" on page 10.](#)

### 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. For EDC 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

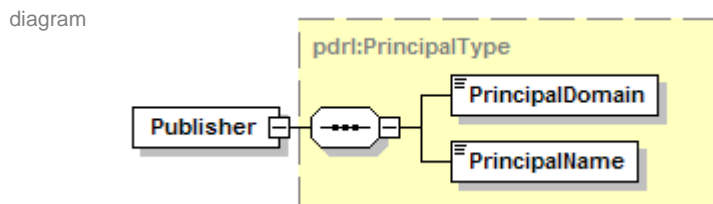
```

source <complexType name="ResourceType">
  <annotation>
    <documentation> Type Definition: Resource is a reference to the
    resource or object for which rights are being granted. For EDC 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 Resource Type/Publisher



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

type `pdrl:PrincipalType`

children `PrincipalDomain PrincipalName`

attributes	Name	Type	Use	Default	Fixed
	PrincipalNameType	pdrl: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 Resource Type/PublishTime



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

type `dateTime`

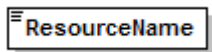
```
source <element name="PublishTime" type="dateTime"/>
```

## 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
```

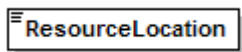
```
source <element name="ResourceName" type="string" minOccurs="0"/>
```

## 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
```

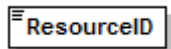
```
source <element name="ResourceLocation" type="anyURI"/>
```

## 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

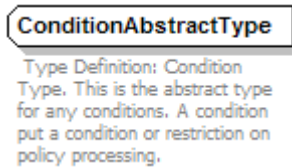
This section defines the condition schema. First, the base classes are defined. Then, the lists of conditions defined within the core schema are itemized.

## 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`

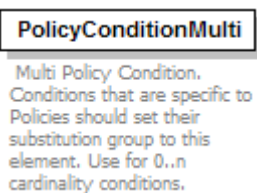
```
source <complexType name="ConditionAbstractType" abstract="true">
  <annotation>
    <documentation> Type Definition: Condition Type. This is the abstract
      type for any conditions. A condition put a condition or restriction on
      policy processing.</documentation>
    </annotation>
  </complexType>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

## PolicyCondition and PolicyConditionEntry Base Elements

The following types are used as base classes for conditions within policies. See [“Conditions” on page 16](#) for more details on conditions.

### element PolicyConditionMulti

diagram



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

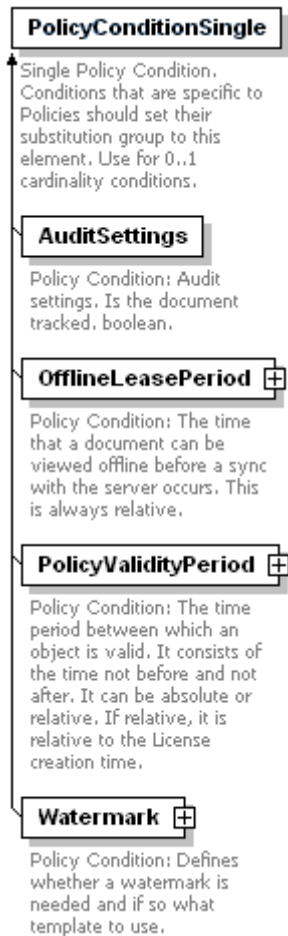
type `pdrl:ConditionAbstractType`

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`

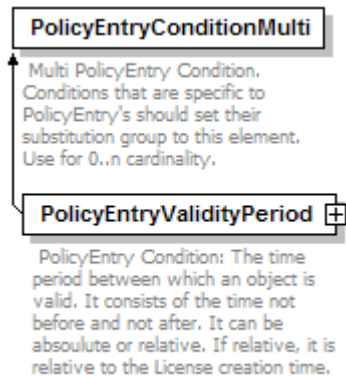
type `pdrl:ConditionAbstractType`

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`

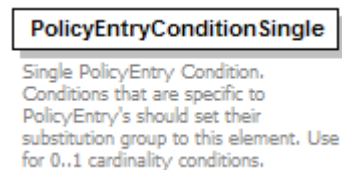
type `pdrl:ConditionAbstractType`

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`

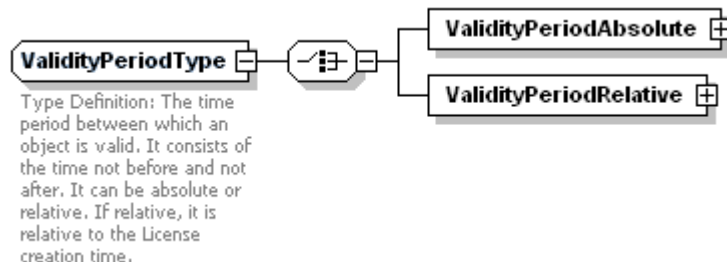
```
source <element name="PolicyEntryConditionSingle"
  type="pdrl:ConditionAbstractType">
  <annotation>
    <documentation>Single PolicyEntry Condition. Conditions that are
      specific to PolicyEntry's should set their substitution group to this
      element. Use for 0..1 cardinality conditions.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

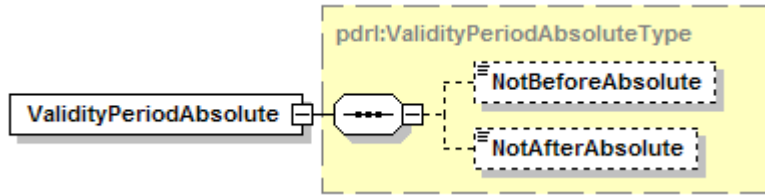
attributes	Name	Type	Use	Default	Fixed
	<code>isAbsoluteTime</code>	<code>boolean</code>	<code>required</code>		

```

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`

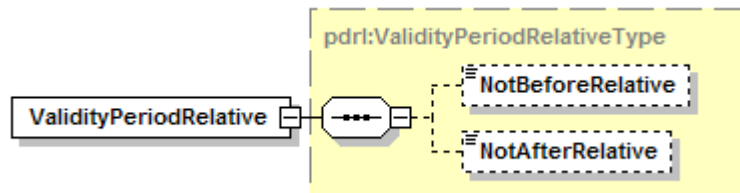
type `pdrl:ValidityPeriodAbsoluteType`

children `NotBeforeAbsolute NotAfterAbsolute`

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

## element ValidityPeriodType/ValidityPeriodRelative

diagram



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

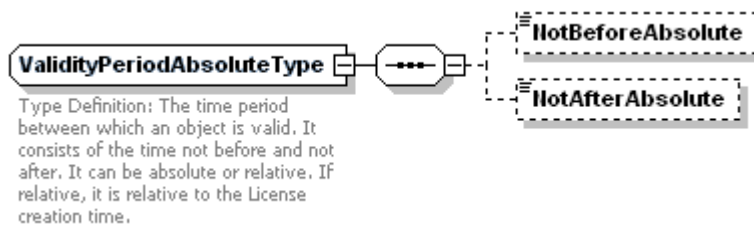
type `pdrl:ValidityPeriodRelativeType`

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



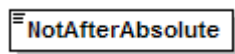
namespace <http://www.adobe.com/schema/1.0/pdrl>

type dateTime

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

## element ValidityPeriodAbsoluteType/NotAfterAbsolute

diagram



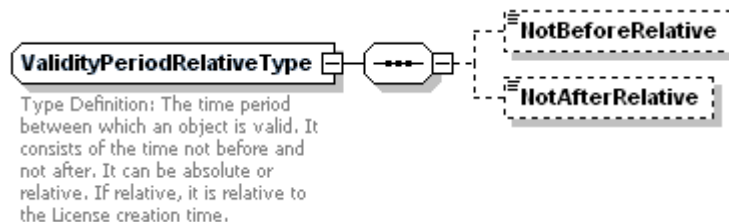
namespace <http://www.adobe.com/schema/1.0/pdrl>

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

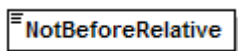
```

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

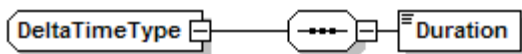
```
source <element name="NotAfterRelative" type="duration" minOccurs="0"/>
```

## 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



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.

namespace <http://www.adobe.com/schema/1.0/pdrl>

type extension of `pdrl:ConditionAbstractType`

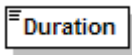
children Duration

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

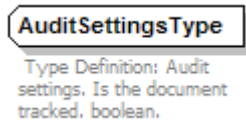
*AuditSettings* is a policy condition that defines whether a document should be tracked. If the *AuditSettings* condition is not present within the policy, the document secured by that policy will not be tracked.

### 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	Default	Fixed
	<code>isTracked</code>	<code>boolean</code>	<code>required</code>		

```

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>

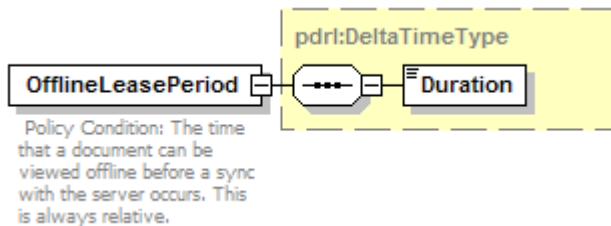
```

## 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>

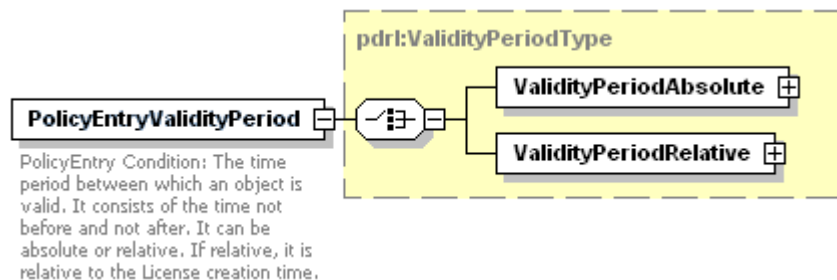
```

## 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” on page 13](#) for details.

### element PolicyEntryValidityPeriod

diagram



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

type `pdrl:ValidityPeriodType`

children `ValidityPeriodAbsolute ValidityPeriodRelative`

attributes	Name	Type	Use	Default	Fixed
	<code>isAbsoluteTime</code>	<code>boolean</code>	<code>required</code>		

```

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>

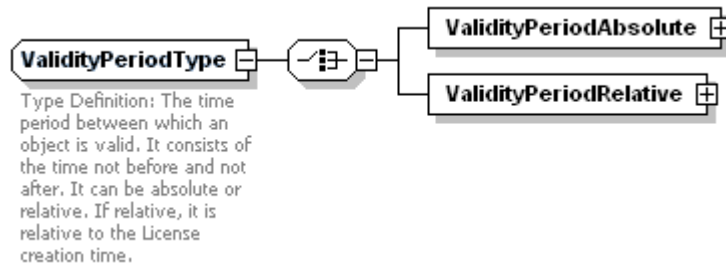
```

## 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” on page 13](#) for details.

## element PolicyValidityPeriod

diagram



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

type `pdrl:ValidityPeriodType`

children `ValidityPeriodAbsolute` `ValidityPeriodRelative`

attributes	Name	Type	Use	Default	Fixed
	<code>isAbsoluteTime</code>	<code>boolean</code>	<code>required</code>		

```
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

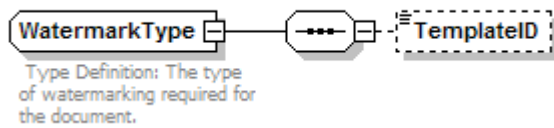
*Watermark* defines whether a watermark is needed and, if so, what template to use. The watermark is identified in the *TemplateID* element using the watermark's unique ID. If this condition is not present, no watermark is applied to the document.

### 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

namespace `http://www.adobe.com/schema/1.0/pdrl`type extension of `pdrl:ConditionAbstractType`children `TemplateID`used by element `Watermark`

attributes	Name	Type	Use	Default	Fixed
	<code>isWatermarked</code>	<code>boolean</code>	<code>required</code>		

```
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"/>
```

This chapter describes conditions and permissions that are specific to Acrobat and LiveCycle Policy Server. See [“Core Schema” on page 18](#) and [“Extensibility” on page 15](#) 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:

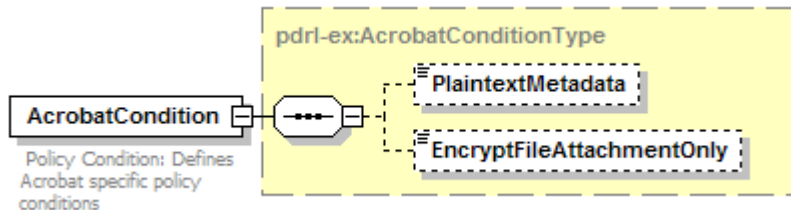
**PlaintextMetadata:** Boolean type. If `TRUE`, metadata is stored as clear-text in the document. If `FALSE` or if not defined, metadata is stored encrypted in the document.

**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`

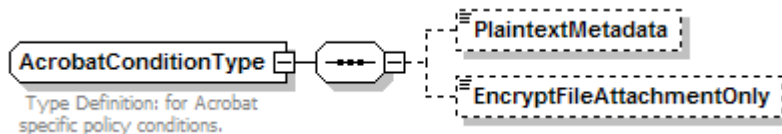
type `pdrl-ex:AcrobatConditionType`

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`

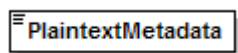
```

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`

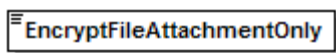
```

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 LiveCycle Policy Server Permissions

This list of permissions are enforced by Acrobat and LiveCycle Policy Server. The definition of these permissions and how they map to Acrobat permissions are found in the Acrobat Permissions Engineering Specification [A-PERM]. They are also provided in the following table.

### simpleType PermissionNameEnum

namespace	http://www.adobe.com/schema/1.0/pdrl-ex	
type	restriction of QName	
facets	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.
	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 integrity protect 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.

Currently, the policy schema does not include integrity protection elements. LiveCycle Policy Server implementation does create an HMAC for the policy and stores the value in an EDC specific portion of the PDF file. To make PDRL more portable, it is suggested that the above elements also be included within the *policy* element in future revisions of PDRL.

### 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 make sure to 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 to 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.

The following documents are referenced in this document.

- PDF      *PDF Reference, fifth edition, Adobe Portable Document Format, Version 1.6*,  
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/xmlsig-core/>
- A-PERM   *Acrobat Permissions Engineering Specification*, Jim Pravetz, Koichi Yoshimura,  
Adobe Systems Incorporated. This document is an internal Adobe specification.