

Information technology — Multimedia framework (MPEG-21) —

Part 5: Rights Expression Language

ICS 35.040

National foreword

This British Standard reproduces verbatim ISO/IEC 21000-5:2004 and implements it as the UK national standard.

The UK participation in its preparation was entrusted to Technical Committee IST/37, Coding of picture, audio, multimedia and hypermedia information, which has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request to its secretary.

Cross-references

The British Standards which implement international publications referred to in this document may be found in the *BSI Catalogue* under the section entitled “International Standards Correspondence Index”, or by using the “Search” facility of the *BSI Electronic Catalogue* or of British Standards Online.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, the ISO/IEC title page, pages ii to ix, a blank page, pages 1 to 121 and a back cover.

The BSI copyright notice displayed in this document indicates when the document was last issued.

Amendments issued since publication

Amd. No.	Date	Comments

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 5 April 2004

© BSI 5 April 2004

ISBN 0 580 43595 4

INTERNATIONAL STANDARD

ISO/IEC 21000-5

First edition
2004-04-01

Information technology — Multimedia framework (MPEG-21) —

Part 5: Rights Expression Language

*Technologies de l'information — Cadre multimédia (MPEG-21) —
Partie 5: Langage d'expression des droits*



Contents

Page

Foreword	viii
Introduction	ix
1 Scope	1
2 Normative references	1
3 Terms, definitions, symbols, and abbreviated terms	2
4 Namespaces and conventions	6
4.1 Namespaces	6
4.2 Namespace conventions	7
4.3 Special typographical conventions	7
4.4 XML structural notation conventions	8
4.5 Authorization context conventions	8
5 Authorization model	8
5.1 General	8
5.2 Authorization request	9
5.3 Authorization context	10
5.4 Authorization story	10
5.5 Authorizer	10
5.6 Authorization model authorization context properties	10
5.7 Authorization proof	11
5.8 License Semantics	12
6 Architectural concepts	12
6.1 Equality	12
6.2 License syntax	12
6.3 License parts	12
6.4 License issuers	13
6.5 Variables	14
6.5.1 Variable definition	14
6.5.2 Variable referencing	14
6.6 Conceptually abstract elements and types	15
6.7 Revocable	15
6.8 Principal surpassing	15
6.8.1 General	15
6.8.2 Principal set representation	15
6.9 Derived grants and grant groups	15
6.9.1 General	15
6.9.2 One-step-derived grants and grant groups	15
6.10 Encrypted Elements	17
7 Core	17
7.1 General	17
7.2 Core authorization context properties	17
7.3 General core elements and types	18
7.3.1 License	18
7.3.2 LicenseGroup	20
7.3.3 ForAll	20
7.3.4 DelegationControl	20
7.3.5 NonSecureReference	21
7.3.6 EncryptedContent	21
7.4 Core conceptually abstract elements and types	21
7.4.1 LicensePart	21

7.4.2	Principal	21
7.4.3	Right	21
7.4.4	Resource	21
7.4.5	Condition	22
7.4.6	AnXmlPatternAbstract	22
7.4.7	DcConstraint	22
7.4.8	TrustRoot	23
7.4.9	ServiceDescription	23
7.4.10	PropertyAbstract	23
7.5	Core principals	23
7.5.1	AllPrincipals	23
7.5.2	KeyHolder	23
7.6	Core rights	24
7.6.1	Issue	24
7.6.2	Obtain	24
7.6.3	PossessProperty	24
7.6.4	Revoke	25
7.7	Core resources	25
7.7.1	DigitalResource	25
7.7.2	Grant	26
7.7.3	GrantGroup	26
7.7.4	Revocable	26
7.7.5	ServiceReference	27
7.8	Core conditions	27
7.8.1	AllConditions	27
7.8.2	ExerciseMechanism	28
7.8.3	ExistsRight	28
7.8.4	Fulfiller	29
7.8.5	PrerequisiteRight	29
7.8.6	RevocationFreshness	29
7.8.7	ValidityInterval	30
7.9	Core patterns	30
7.9.1	General patterns	30
7.9.2	Principal patterns – PropertyPossessor	31
7.9.3	Right patterns	31
7.9.4	Resource patterns	31
7.9.5	Condition patterns	33
7.10	Core delegation constraints	33
7.10.1	ConditionIncremental	33
7.10.2	ConditionUnchanged	33
7.10.3	DepthConstraint	33
7.10.4	ToConstraint	34
7.11	Core trust roots	35
7.11.1	TrustedRootGrants	35
7.11.2	TrustedRootIssuers	35
7.12	Core service descriptions	35
7.13	Core properties	35
8	Standard extension	35
8.1	General	35
8.2	Standard extension authorization context properties	35
8.3	General standard extension elements and types	38
8.3.1	AccountPayable	38
8.3.2	ProfileCompliance	38
8.3.3	Rate	39
8.3.4	StateDistinguisher	39
8.3.5	UddiKey	39
8.3.6	Uuid	39
8.4	Standard extension conceptually abstract elements and types	39
8.4.1	Name	39

8.4.2	StatefulCondition	39
8.5	Standard extension principals	39
8.6	Standard extension rights – RightUri	40
8.7	Standard extension resources	40
8.8	Standard extension conditions	40
8.8.1	CallForCondition	40
8.8.2	ExerciseLimit	40
8.8.3	FeeFlat	40
8.8.4	FeeMetered	41
8.8.5	FeePerInterval	41
8.8.6	FeePerUse	42
8.8.7	FeePerUsePrePay	42
8.8.8	SeekApproval	42
8.8.9	Territory	43
8.8.10	TrackQuery	43
8.8.11	TrackReport	43
8.8.12	TransferControl	44
8.8.13	ValidityIntervalFloating	44
8.8.14	ValidityIntervalStartsNow	44
8.8.15	ValidityTimeMetered	45
8.8.16	ValidityTimePeriodic	45
8.9	Standard extension patterns	46
8.9.1	General patterns	46
8.9.2	Principal patterns	46
8.9.3	Right patterns	46
8.9.4	Resource patterns – X509SubjectNamePattern	47
8.9.5	Condition patterns – ValidityIntervalDurationPattern	47
8.10	Standard extension delegation constraints	47
8.11	Standard extension trust roots	47
8.12	Standard extension service descriptions	47
8.12.1	AnonymousStateService	47
8.12.2	Uddi	48
8.12.3	WsdIAddress	48
8.12.4	WsdIComplete	49
8.13	Standard extension properties – PropertyUri	49
8.14	Standard extension name properties	49
8.14.1	CommonName	49
8.14.2	DnsName	49
8.14.3	EmailName	49
8.14.4	X509SubjectName	49
9	Multimedia extension	50
9.1	General	50
9.2	Multimedia extension authorization context properties	50
9.3	General multimedia extension elements and types – resource attribute set definitions	51
9.3.1	Complement	51
9.3.2	Intersection	51
9.3.3	Set	52
9.3.4	Union	52
9.4	Multimedia extension conceptually abstract elements and types	52
9.5	Multimedia extension principals	52
9.6	Multimedia extension rights	52
9.6.1	Adapt	52
9.6.2	Delete	53
9.6.3	Diminish	53
9.6.4	Embed	53
9.6.5	Enhance	53
9.6.6	Enlarge	54
9.6.7	Execute	54
9.6.8	Install	54

9.6.9	Modify	54
9.6.10	Move	55
9.6.11	Play	55
9.6.12	Print	55
9.6.13	Reduce.....	55
9.6.14	Uninstall	56
9.7	Multimedia extension resources.....	56
9.7.1	DiItemReference	56
9.7.2	DiReference.....	56
9.8	Multimedia extension conditions.....	56
9.8.1	Digital Item conditions	56
9.8.2	Marking conditions.....	57
9.8.3	Security conditions	57
9.8.4	Transaction	59
9.8.5	Resource attribute conditions	59
9.9	Multimedia extension patterns.....	60
9.10	Multimedia extension delegation constraints	60
9.11	Multimedia extension trust roots.....	60
9.12	Multimedia extension service descriptions	60
9.13	Multimedia extension properties	60
9.14	Multimedia extension name properties.....	60
Annex A	(normative) W3C XML Schemas	61
A.1	General	61
A.2	Schema for the Core Namespace	61
A.3	Schema for the Standard Extension Namespace	72
A.4	Schema for the Multimedia Extension Namespace	80
Annex B	(normative) Country, region, and currency Qualified Names.....	87
B.1	Namespace URI structure.....	87
B.2	Country Qualified Names	87
B.3	Region Qualified Names	87
B.4	Currency Qualified Names	88
Annex C	(informative) Simplified equality algorithm	89
C.1	General	89
C.2	The equalQuickItem function	89
C.3	The equalQuickList function	91
C.4	The equalQuickSimple function.....	91
Annex D	(informative) Example Rights Expressions	92
D.1	Overview of examples.....	92
D.2	Simple end-user License example.....	92
D.3	Distribution License example	94
Annex E	(informative) Design philosophy concerning extensions and profiles of this part of ISO/IEC 21000	99
E.1	General	99
E.2	Definition of extension.....	99
E.3	Definition of profile.....	99
E.4	Extensibility points.....	99
Annex F	(informative) Extension mechanisms for introducing new rights	101
F.1	General	101
F.2	Use existing rights and conditions.....	101
F.3	Use rightUri	102
F.4	Use type extension.....	103
F.5	Use element extension	103
Annex G	(informative) Example profile of this part of ISO/IEC 21000	104
G.1	General	104
G.2	Profile elements.....	104
G.3	Profile schema	105

G.3.1	General	105
G.3.2	Example profile schema for the Core Namespace	105
G.3.3	Example profile schema for the Standard Extension Namespace.....	106
G.3.4	Example profile schema for the Multimedia Extension Namespace	107
G.4	Profile signalling	107
G.5	Profile expressiveness	107
G.5.1	General	107
G.5.2	Everyone can play a song for the year of 2004.....	107
G.5.3	A key holder can play a song three times within the year of 2004	108
Annex H	(informative) Relationship between ISO/IEC 21000-6 and this part of ISO/IEC 21000	110
Annex I	(informative) Relationship between ISO/IEC 21000-2 and this part of ISO/IEC 21000	111
Annex J	(informative) Example revocation mechanism	113
J.1	General	113
J.2	Designing a revocation mechanism.....	113
J.3	Indicating a revocation mechanism and freshness Condition when issuing a License	114
J.4	Checking for freshness	115
J.5	Delegating revoke Right	116
J.6	Requesting revocation	117
J.7	Latency until revocation goes into effect	119
J.8	Latency until freshness duration passes	119
Annex K	(informative) Patent statements	120
Bibliography		121

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 21000-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 21000 consists of the following parts, under the general title *Information technology — Multimedia framework (MPEG-21)*:

- *Part 1: Vision, Technologies and Strategy* [Technical Report]
- *Part 2: Digital Item Declaration*
- *Part 3: Digital Item Identification*
- *Part 5: Rights Expression Language*
- *Part 6: Rights Data Dictionary*
- *Part 7: Digital Item Adaptation*

Introduction

The growth of the Internet has enabled worldwide distribution and consumption of valuable multimedia resources, reduced the cost of doing business, enabled new business models for industry participants, and provided consumers with unprecedented access to high-quality multimedia resources. Internal distribution, external distribution, and retail sales now are conducted on the Internet to establish cost effective, reliable, flexible, highly available, and secure means of managing the delivery of multimedia resources. Consumers routinely search for and download multimedia resources from sources world-wide and can conveniently redistribute those resources. This has fuelled the development of technologies to manage, secure, control, and automate the flow of multimedia resources over the Internet.

The free and convenient flow of multimedia resources through the Internet presents many challenges to content owners and distributors. Before making high-quality and valuable multimedia resources available online, content owners want to be assured that their rights to those resources are respected. In addition, the business models and contracts of content distributors often involve conditions regarding distribution, such as fees, territory restrictions, time limits, and so on.

To meet these requirements, the players involved in the online distribution and consumption of multimedia resources need to exchange information about the rights, terms, and conditions associated with each resource at each step in the multimedia resource lifecycle. For example, a publisher needs to communicate the available consumption rights and the terms and conditions under which those rights may be exercised. To use the multimedia resources, a consumer needs to know the types of usage allowed and the terms and conditions that must be met. In distribution and super distribution business models, this information needs to be communicated to each participant in the distribution chain.

Depending on the business model, expressing rights, terms, and conditions can be simple or complex. In a simple example, a consumer might pay a flat fee to obtain unlimited rights to play a video file. In a more complex example, a video publisher might grant a distributor the right to sell usage rights for classic movie titles to consumers. The distribution agreement might specify the rights that consumers may purchase, the maximum fee the distributor may charge, and a percentage of the fee that must be paid to the publisher.

In an end-to-end system, other considerations such as authenticity and integrity of Rights Expressions become important. For example, any party who issues rights to use or distribute multimedia resources must be identified and authorized. In addition, a Rights Expression may be accessed by different participants during its life cycle, which requires mechanisms and semantics for validating the authenticity and integrity of the Rights Expression.

To address many of these issues, a common Rights Expression Language that can be shared among all participants in this digital workflow is required. A common Rights Expression Language addresses important aspects of the interoperability issues inherent in digital multimedia resource distribution; the issues relating to exchanging Rights Expressions during their life cycle; and the system issues such as trust, authorization, and authentication.

This part of ISO/IEC 21000 addresses a part of the overall vision for ISO/IEC 21000, which is to define a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities.

Information technology — Multimedia framework (MPEG-21) —

Part 5: Rights Expression Language

1 Scope

This part of ISO/IEC 21000 specifies the syntax and semantics of a Rights Expression Language.

This part of ISO/IEC 21000 does not give any permission, including permissions about who is legally or technically allowed to create Rights Expressions. It does not specify the security measures of trusted systems, propose specific applications, or describe the details of the systems required for accounting (monetary transactions, state transactions, and so on). It also does not specify if or when Rights Expressions shall be consulted.

However, this part of ISO/IEC 21000 does define an authorization model to specify whether the semantics of a set of Rights Expressions permit a given Principal to perform a given Right upon a given optional Resource during a given time interval based on a given authorization context and a given trust root.

Clause 1 gives the scope of this part of ISO/IEC 21000. Clause 2 gives the normative references. Clause 3 gives pertinent terms, definitions, symbols, and abbreviated terms. Clause 4 gives the namespaces and conventions. Clause 5 specifies the authorization model. Clause 6 defines architectural concepts. Clause 7 specifies the syntax and semantics integral to the architecture. Clause 8 specifies syntax and semantics peripheral to the architecture but still useful in many domains beyond multimedia. Clause 9 specifies syntax and semantics specific to multimedia. Annex A uses W3C XML Schema to normatively specify the syntax of the types and elements defined throughout this part of ISO/IEC 21000. Annex B normatively defines Qualified Names for identifying countries, regions, and currencies. Annex C gives an informative simplified equality algorithm. Annex D gives some example Rights Expressions. Annex E describes the design philosophy for extensions and profiles of this part of ISO/IEC 21000. Annex F demonstrates how to introduce new rights as an extension to this part of ISO/IEC 21000. Annex G gives an example profile of this part of ISO/IEC 21000. Annex H describes the relationship between ISO/IEC 21000-6 and this part of ISO/IEC 21000. Annex I describes the relationship between ISO/IEC 21000-2 and this part of ISO/IEC 21000. Annex J describes an example revocation mechanism and gives a walk-through of revocation.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 3166 (all parts), *Codes for the representation of names of countries and their subdivisions*

ISO 4217, *Codes for the representation of currencies and funds*

ISO/IEC 9594-8, *Information technology — Open Systems Interconnection — The Directory: Public-key and attribute certificate frameworks*

ISO/IEC 10021-2, *Information technology — Message Handling Systems (MHS): Overall architecture*

ISO/IEC 21000 (all parts), *Information technology — Multimedia framework (MPEG-21)*

Request for Comments (RFC) 1034, *Domain Names — Concepts and Facilities*, The Internet Engineering Task Force (IETF), November 1987, available at <<http://www.ietf.org/rfc/rfc1034>>

RFC 1738, *Uniform Resource Locators (URL)*, IETF, December 1994, available at <<http://www.ietf.org/rfc/rfc1738.txt>>

RFC 2141, *URN Syntax*, IETF, May 1997, available at <<http://www.ietf.org/rfc/rfc2141.txt>>

RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*, IETF, August 1998, available at <<http://www.ietf.org/rfc/rfc2396.txt>>

RFC 2822, *Internet Message Format*, IETF, April 2001, available at <<http://www.ietf.org/rfc/rfc2822>>

ROUTINGABA, *Routing Number Policy*, American Bankers Association Routing Number Administrative Board, November 1996, available at <http://www.tfp.com/text/aba_policy.pdf>

SCC14N, *Schema Centric XML Canonicalization Version 1.0*, Organization for the Advancement of Structured Information Standards (OASIS) Universal Description, Discovery, and Integration (UDDI) Technical Committee (TC) Committee Specification, 19 July 2002, available at <<http://www.uddi.org/pubs/SchemaCentricCanonicalization-20020710.htm>>

UDDIV2API, *UDDI Version 2.04 API Specification*, OASIS Standard, 19 July 2002, available at <<http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm>>

UDDIV2DSR, *UDDI Version 2.03 Data Structure Reference*, OASIS Standard, 19 July 2002, available at <<http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm>>

UDDIV3, *UDDI Version 3.0*, OASIS UDDI TC Committee Specification, 19 July 2002, available at <<http://uddi.org/pubs/uddi-v3.00-published-20020719.htm>>

WSDLSPEC, *Web Services Description Language (WSDL) 1.1*, Worldwide Web Consortium (W3C) Note, 15 March 2001, available at <<http://www.w3.org/TR/2001/NOTE-wsdl-20010315>>

XMLSPEC, *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, 6 October 2002, available at <<http://www.w3.org/TR/2000/REC-xml-20001006>>

XMLDSIG, *XML-Signature Syntax and Processing*, W3C Recommendation, 12 February 2002, available at <<http://www.w3.org/TR/2002/REC-xmlsig-core-20020212>>

XMLENC, *XML-Encryption Syntax and Processing*, W3C Recommendation, 10 December 2002, available at <<http://www.w3.org/TR/2002/REC-xmlenc-core-20021210>>

XMLINFOSET, *XML Information Set*, W3C Recommendation, 24 October 2001, available at <<http://www.w3.org/TR/2001/REC-xml-infoset-20011024>>

XMLNAMES, *Namespaces in XML*, W3C Recommendation, 14 January 1999, available at <<http://www.w3.org/TR/1999/REC-xml-names-19990114>>

XMLSCHEMA, *XML Schema Part 1: Structures and Part 2: Datatypes*, W3C Recommendation, 2 May 2001, available at <<http://www.w3.org/TR/2001/REC-xmlschema-1-20010502>> and <<http://www.w3.org/TR/2001/REC-xmlschema-2-20010502>>

XPATH, *XML Path Language (XPath) Version 1.0*, W3C Recommendation, 16 November 1999, available at <<http://www.w3.org/TR/1999/REC-xpath-19991116>>

3 Terms, definitions, symbols, and abbreviated terms

For the purposes of this document, the following terms, definitions, symbols, and abbreviated terms apply.

3.1

Allowable Destination Condition

allowable destination condition as mentioned in 7.4.7

3.2**Allowable Destination Delegation Control**

allowable destination delegation control as mentioned in 7.4.7

3.3**Allowable Destination Principal**

allowable destination principal as mentioned in 7.4.7

3.4**authorization context**

set of properties having the characteristics defined in 5.3

3.5**Authorization Context Member**

fifth member

NOTE This term is used with respect to an authorization request, as in “the Authorization Context Member of the authorization request.”

3.6**authorization proof**

authorization story having the characteristics defined in 5.7

3.7**authorization request**

ordered seven-tuple as defined in 5.2

3.8**authorization story**

ordered triple as defined in 5.4

3.9**authorizer**

ordered five-tuple as defined in 5.5

3.10**Business Service**

businessService as defined in UDDIV2DSR and UDDIV3

3.11**conceptually abstract**

designated as being subject to the provisions given in 6.6

3.12**Condition**

permission limitation identified by an `r:Condition`

NOTE See 4.3 for the meaning of `r:Condition`.

3.13**Conditionally**

in a manner subject to a Condition

3.14**Core Namespace**

Namespace for the names defined in Clauses 5, 6, and 7

3.15**Derived**

derived as defined in 6.9.1

3.16

Equal

equal as defined in 6.1

3.17

License

Rights Expression

expression that is created by Principals to Conditionally or Unconditionally permit the same or other Principals to perform Rights upon Resources

3.18

Match

match as mentioned in 7.4.6.1

3.19

Multimedia Extension Namespace

Namespace for the names defined in Clause 9

3.20

Namespace

XML namespace as defined in XMLNAMES

3.21

performance

carrying out or execution

3.22

primitive grant

`r:Grant` that has no child `r:forAll` element

3.23

Principal

system entity identified by an `r:Principal`

NOTE 1 See 4.3 for the meaning of `r:Principal`.

NOTE 2 Many Users (as introduced in ISO/IEC 21000-1) are Principals.

3.24

Qualified Name

qualified name as defined in XMLNAMES

3.25

Registry

Universal Description, Discovery, and Integration registry as defined in UDDIV2API and UDDIV3

3.26

repository

Principal that can hold Resources

EXAMPLE Personal systems, on-line storefronts, libraries, and archives are examples of repositories.

3.27

Resource

entity, quality, event, state, concept, substance, or anything else referred to by a noun and identified by an `r:Resource`

NOTE 1 See 4.3 for the meaning of `r:Resource`.

NOTE 2 ISO/IEC 21000-2 defines *resource* in a narrower sense than Resource is defined here. The relation might be characterized by the expression "a *resource* is a Resource that is also an asset."

3.28

resource attribute set definition element

element that is one of those defined in 9.3

3.29

Resource Member

third member

NOTE This term is used with respect to an authorization request, as in "the Resource Member of the authorization request."

3.30

revocable

ordered pair as defined in 6.7

3.31

Right

act identified by an *r:Right*

NOTE See 4.3 for the meaning of *r:Right*.

3.32

Right Member

second member

NOTE This term is used with respect to an authorization request, as in "the Right Member of the authorization request."

3.33

Satisfied

satisfied as mentioned in 7.4.5

3.34

Standard Extension Namespace

Namespace for the names defined in Clause 8

3.35

Surpasses

surpasses as defined in 6.8.1

3.36

service

system entity that provides functionality

NOTE 1 There is no requirement that a service be located on a physically different machine than a client.

NOTE 2 There is no requirement that a service be part of a different executable than a client.

3.37

system entity

entity that is capable of acting in a system

EXAMPLE An automated process, a subsystem, and a person or group of persons are examples of system entities.

3.38
Trust Root Member
seventh member

NOTE This term is used with respect to an authorization request, as in "the Trust Root Member of the authorization request."

3.39
Unconditionally
in a manner not subject to any Condition

3.40
URI
Uniform Resource Identifier

[RFC 2396]

3.41
Variable
variable as defined in 6.5.1.1

3.42
WSDL
Web Services Description Language

[WSDLSPEC]

3.43
XML
Extensible Markup Language

[XMLSPEC]

3.44
XPath
XML Path Language

[XPATH]

3.45
I
the `r:issue` element

3.46
P
the `r:possessProperty` element

4 Namespaces and conventions

4.1 Namespaces

The Core Namespace shall be `urn:mpeg:mpeg21:2003:01-REL-R-NS`. The Standard Extension Namespace shall be `urn:mpeg:mpeg21:2003:01-REL-SX-NS`. The Multimedia Extension Namespace shall be `urn:mpeg:mpeg21:2003:01-REL-MX-NS`. In each of these, the 01 represents a serial number that is expected to change as this part of ISO/IEC 21000 evolves.

4.2 Namespace conventions

Throughout this part of ISO/IEC 21000, Qualified Names are written with a namespace prefix followed by a colon followed by the local part of the Qualified Name as shown in the following example.

EXAMPLE `r:grant`

For clarity, throughout this part of ISO/IEC 21000, consistent namespace prefixes are used. Table 1 gives these prefixes and the corresponding namespace. Licenses compliant with this part of ISO/IEC 21000 are not limited to the namespace prefixes in Table 1 for the corresponding namespace and may use any appropriate namespace prefix that is declared in the License.

Table 1 — Namespace prefixes

Namespace prefix	Namespace
<code>r</code>	<code>urn:mpeg:mpeg21:2003:01-REL-R-NS</code>
<code>sx</code>	<code>urn:mpeg:mpeg21:2003:01-REL-SX-NS</code>
<code>mx</code>	<code>urn:mpeg:mpeg21:2003:01-REL-MX-NS</code>
<code>dsig</code>	<code>http://www.w3.org/2000/09/xmldsig#</code>
<code>xenc</code>	<code>http://www.w3.org/2001/04/xmlenc#</code>
<code>xsd</code>	<code>http://www.w3.org/2001/XMLSchema</code>
<code>xsi</code>	<code>http://www.w3.org/2001/XMLSchema-instance</code>
<code>didl</code>	<code>urn:mpeg:mpeg21:2002:01-DIDL-NS</code>
<code>dii</code>	<code>urn:mpeg:mpeg21:2002:01-DII-NS</code>
<code>wsdl</code>	<code>http://schemas.xmlsoap.org/wsdl/</code>
<code>soap</code>	<code>http://schemas.xmlsoap.org/wsdl/soap/</code>
<code>acme</code>	This namespace prefix is used for demonstration only.

4.3 Special typographical conventions

Fixed-width font is used to indicate literal machine-readable character sequences.

The names of XML Schema elements and attributes appear in fixed-width font in mixed case with an initial lower case letter. The names of XML Schema types appear in fixed-width font in mixed case with an initial upper case letter.

Specific linguistic constructions are used in this part of ISO/IEC 21000 with respect to these conventions in order to more concisely convey the intention. They are as follows:

- A passage that mentions an element, as in "the `r:grant`," is using the word in a technical sense to refer to the notion of that element as an XML Schema element.
- A passage that mentions a type and prefixes the type with the word *type*, as in "the type `r:Grant`," is using the word in a technical sense to refer to the notion of that type as an XML Schema type.
- A passage that mentions a type and prefixes the type with an article, such as *a* or *the*, as in "an `r:Grant`," is using the word in a technical sense to refer to any element whose type is that type or any derivation thereof (or, in the case of a simple type, any simple content whose type is that simple type or any derivation thereof). Semantics assigned to a type in this way shall not be overridden by type derivations or elements using the type; type derivations or elements that use the type may alter the semantics only as long as all the statements made about the type in these passages still hold for the type derivations and elements that use the type.

4.4 XML structural notation conventions

To refer to child elements of an element represented by a variable, the variable name is followed by a forward slash and the name of the particle against which the child elements validate as shown in Example 1.

EXAMPLE 1 Suppose g is the following $r:grant$:

```
<r:grant>
  <r:keyHolder>...</r:keyHolder>
  <acme:extensionRight/>
  <r:keyHolder>...</r:keyHolder>
</r:grant>
```

In this case, $g/r:principal$ would refer to the first $r:keyHolder$ child because it validates against the $r:principal$ particle, which comes before the $r:right$ particle in an $r:grant$. The second $r:keyHolder$ validates against the $r:resource$ particle, which comes after the $r:right$ in an $r:grant$.

NOTE For more details about the syntax of $r:grant$, see Annex A.

To refer to child elements of child elements of an element represented by a variable, another forward slash is used as shown in Example 2.

EXAMPLE 2 $c/sx:location/sx:country$

To refer to attributes of an element represented by a variable, the variable name is followed by a forward slash, an at sign (@), and then the name of the attribute as shown in Example 3.

EXAMPLE 3 $//@r:licenseId$

To avoid confusion with XPath notation, when XPath notation is used in this part of ISO/IEC 21000, its use is clearly indicated with wording such as "the following XPath location path."

4.5 Authorization context conventions

To refer to the value of a property of an authorization context variable, the variable name is followed by a full stop followed by the property name as shown in Example 1 and Example 2.

EXAMPLE 1 $\Sigma.acme:a1()$ refers to the value of the $acme:a1$ property of the authorization context Σ .

EXAMPLE 2 If the value of x is 123, then $\Sigma.acme:a2(x)$ refers to the value of the $acme:a2(123)$ property of the authorization context Σ .

NOTE For more details about authorization contexts, see 5.3.

5 Authorization model

5.1 General

This Clause specifies the authorization model.

NOTE The authorization model makes use of an authorization request, an authorization context, an authorization story, and an authorizer. These are used to create authorization proofs which help to define the semantics of Licenses with respect to authorization requests. Figure 1 illustrates the relationship between the components of the authorization model.

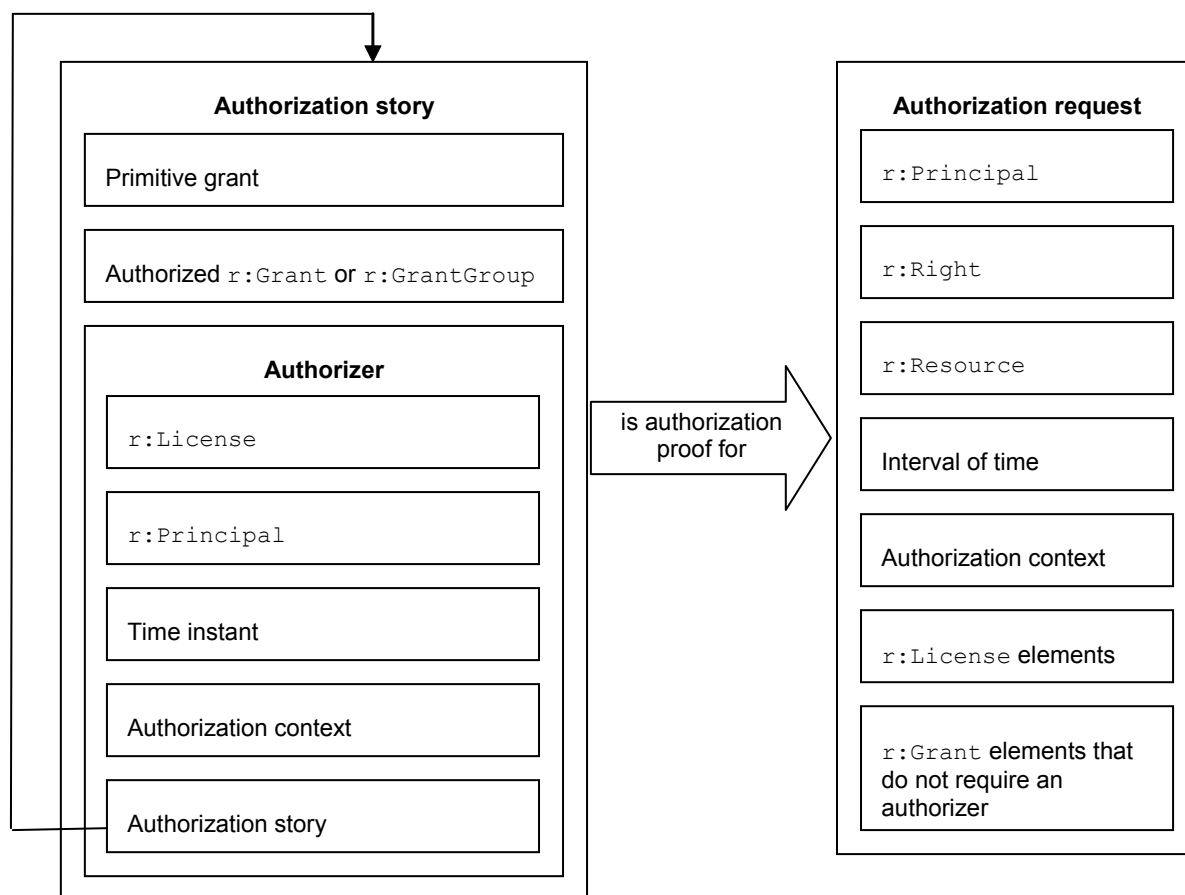


Figure 1 — Authorization Model

5.2 Authorization request

An authorization request is an ordered seven-tuple containing the following members, in order:

- optionally, the `r:Principal` identifying the Principal for which permission is requested,
- the `r:Right` identifying the Right the performance of which is requested to be permitted,
- optionally, the `r:Resource` identifying the Resource upon which permission is requested,
- the interval of time in which the requested performance of b) by a) upon c) is to occur,
- the authorization context containing properties representing statements that are to be considered true for the purposes of establishing the requested permission,
- the set of `r:License` elements that may be consulted to establish the requested permission, and
- the set of `r:Grant` elements that do not require an authorizer for the purposes of establishing the requested permission.

NOTE 1 An authorization request can be conceptualized as representing the question "is it permitted for a given Principal to perform a given Right upon a given Resource during a given time interval based on a given authorization context, a given set of Licenses, and a given trust root?"

Each of the `r:License` elements *l* in the set mentioned in f) shall have exactly one `//r:issuer` child.

NOTE 2 The semantics of an `r:License` / with a different number of `//r:issuer` children is specified in terms of `r:License` elements with one such child each (see 6.4 for the specifics).

5.3 Authorization context

An authorization context is a set of properties having the following characteristics:

- each property consists of exactly one name and one value,
- a property name consists of exactly one Qualified Name and zero or more parameters,
- no two properties in an authorization context share the same property name, and
- each property represents a statement.

NOTE 5.6, 7.2, 8.2, and 9.2 define authorization context properties and the statements they represent.

5.4 Authorization story

An authorization story is an ordered triple containing the following members, in order:

- a) a primitive grant,
- b) either an `r:Grant` or an `r:GrantGroup`, and
- c) optionally, an authorizer.

NOTE Conceptually, the first member of an authorization story is used to demonstrate to which authorization requests the authorization story applies. The second member of an authorization story represents the actual `r:Grant` or `r:GrantGroup` that is authorized by the third member of the authorization story.

5.5 Authorizer

An authorizer is an ordered five-tuple containing the following members, in order:

- a) an `r:License`,
- b) an `r:Principal`,
- c) a time instant,
- d) an authorization context, and
- e) an authorization story.

NOTE Conceptually, if an `r:Grant` or `r:GrantGroup` is authorized, it is part of a License and is authorized in that License by some Principal at some time instant based on some authorization context and there is an authorization story explaining why that Principal was permitted to make that authorization.

5.6 Authorization model authorization context properties

Table 2 specifies the authorization context properties relating specifically to the authorization model and the statements they represent. If a property has the name given in the first column of Table 2 and the value given in the second column of Table 2, then the statement represented by that property is the statement given in the third column of Table 2.

Table 2 — Authorization model authorization context properties

Property name	Property value	Statement represented
<code>r:issueTime(<i>l</i>, <i>p</i>)</code>	<i>i</i>	<i>l</i> is an <code>r:License</code>, <i>p</i> is an <code>r:Principal</code>, <i>i</i> is a time instant, and <i>p</i> issued <i>l</i> at <i>i</i>. NOTE 1 The <code>r:timeOfIssue</code> field in an <code>r:IssuerDetails</code> (see 7.3.1.6.3) can be useful in determining when <i>p</i> issued <i>l</i>. However, it is wise to give consideration as to whether the issuer is trustworthy and, when in doubt, to seek additional proof, such as in the form of signatures and countersignatures. NOTE 2 The <code>r:Issuer</code> can be useful in determining whether <i>p</i> issued <i>l</i>. However, it is wise to give consideration as to whether the information given in the <code>r:Issuer</code> is trustworthy and, when in doubt, to seek additional proof, such as in the form of signatures and countersignatures.
<code>r:issueContext(<i>l</i>, <i>p</i>, <i>h</i>, Σ)</code>	true	<i>l</i> is an <code>r:License</code>, <i>p</i> is an <code>r:Principal</code>, <i>h</i> is either an <code>r:Grant</code> or an <code>r:GrantGroup</code>, Σ is an authorization context, and the statements represented by the properties in Σ are all true for the purposes of establishing the permission for the Principal identified by <i>p</i> to include an <code>r:Grant</code> or <code>r:GrantGroup</code> that is Equal to <i>h</i> as <code>//r:grant</code> or <code>//r:grantGroup</code> when issuing <i>l</i>.

5.7 Authorization proof

A finite authorization story (*g*, *h*, *e*) is said to be an authorization proof for an authorization request (*p*, *r*, *t*, *v*, Σ , *L*, *R*) if and only if all of the following are true:

- either *p* Surpasses `g/r:principal` or `g/r:principal` is absent,
- `g/r:right` is Equal to *r*,
- either `g/r:resource` is Equal to *t* or both are absent,
- either `g/r:condition` is Satisfied with respect to authorization request (*p*, *r*, *t*, *v*, Σ , *L*, *R*) and authorization story (*g*, *h*, *e*) or `g/r:condition` is absent,
- *g* is Derived from *h* with respect to authorization request (*p*, *r*, *t*, *v*, Σ , *L*, *R*) and authorizer *e*,
- if *e* is absent, then *h* is a member of *R*, and
- if *e* is present, then, letting (*l*, π , *i*, σ , *s*) be the ordered five-tuple representation of *e*, all of the following are true:
 - *l* is a member of *L*,
 - *h* is Equal to one of the `//r:Grant` or `//r:GrantGroup` children of *l*,
 - Σ :`issueTime`(*l*, π) is *i*,
 - Σ :`issueContext`(*l*, π , *h*, σ) is true,
 - the time instant *i* is no later in time than the start of time interval *v*, and
 - *s* is an authorization proof for the authorization request (π , *I*, *h*, ψ , σ , *L*, *R*) where ψ is a time interval of zero length starting at *i*.

5.8 License Semantics

If there is an authorization proof for an authorization request $(p, r, t, v, \Sigma, L, R)$, then the semantics of the set of `r:License` elements L with respect to that authorization request is that they permit the optional Principal identified by p to perform the Right identified by r over the optional Resource identified by t during the time interval v based on the authorization context Σ and the set R of `r:Grant` elements that do not require an authorizer.

6 Architectural concepts

6.1 Equality

Let a and b be XML elements. Let α and β be the result of the execution of the Schema Centric Canonicalization algorithm (see SCC14N) on an XML Information Set (see XMLINFOSET) whose document information item contains in its [children] property the item a or b , respectively. Then a is Equal to b if and only if the bits strings α and β are bit-for-bit identical.

NOTE Annex C presents an approach to efficiently testing for equality.

6.2 License syntax

Licenses shall be valid `r:License` elements.

6.3 License parts

The following provisions apply to `r:License` elements that contain `r:LicensePart` elements having the `r:licensePartId` or `r:licensePartIdRef` attributes.

- a) An `r:LicensePart` shall not have both the `r:licensePartId` attribute and the `r:licensePartIdRef` attribute.
- b) For a given `r:LicensePartId` value v and `r:License` l , l shall not contain more than one `r:LicensePart` having an `r:licensePartId` attribute with the value v .
- c) If an `r:LicensePart` has an `r:licensePartIdRef` attribute, then it shall have empty content. As a corollary, types that are derivations of the type `r:LicensePart` should allow their content to be empty (for otherwise their `r:licensePartIdRef` attribute shall not be used).
- d) If an `r:LicensePart` a has an `r:licensePartIdRef` attribute with a certain value v , then there shall exist some (other) `r:LicensePart` b in the same `r:License` as a such that all of the following are true:
 - 1) b has an `r:licensePartId` attribute with value v ,
 - 2) the expanded element name of b exactly matches that of a , and
 - 3) b is not an ancestor of a .
- e) With the exception of signature verification and License issuer resolution (see 6.4), the following two processing steps shall be carried out before the other License processing steps defined in this part of ISO/IEC 21000. In particular, they shall be carried out before the evaluation of Variable references or the testing of equality.
 - 1) If an `r:LicensePart` a has an `r:licensePartIdRef` attribute with a certain value v , and b is the `r:LicensePart` in the same `r:License` as a that has an `r:licensePartId` attribute with value v , then the semantics of the `r:License` containing a and b are as if
 - i) a were removed from the `r:License` and replaced with an element α that is Equal to b ,

- ii) the `r:licensePartId` attributes were removed from α and all of its descendants, and
 - iii) any preserved attributes that α has were removed from α , and any preserved attributes that a has were added to α , where here a preserved attribute is
 - I) any attribute of type `xsd:ID` or
 - II) any attribute for which `id` is the local part of its Qualified Name.
- 2) If an `r:License` contains no `r:LicensePart` elements with an `r:licensePartIdRef` attribute, then the semantics of that `r:License` are as if the `r:licensePartId` attribute were removed from all `r:LicensePart` elements in that `r:License` with an `r:licensePartId` attribute.

NOTE 1 It is the intent of e)1)iii)II) to enable the useful definition and incorporation of other identification systems on `r:LicensePart` elements beyond the document-global `xsd:ID`-typed identifiers.

NOTE 2 Circular references such as that shown in Example 1 are not allowed because, after one application of provision e), the semantics will be as shown in Example 2, which violates provision d)3).

EXAMPLE 1

```
<r:license>
  <r:grantGroup licensePartId="x">
    <r:grantGroup licensePartIdRef="y"/>
  </r:grantGroup>
  <r:grantGroup licensePartId="y">
    <r:grantGroup licensePartIdRef="x"/>
  </r:grantGroup>
</r:license>
```

EXAMPLE 2

```
<r:license>
  <r:grantGroup licensePartId="x">
    <r:grantGroup>
      <r:grantGroup licensePartIdRef="x"/>
    </r:grantGroup>
  </r:grantGroup>
  <r:grantGroup licensePartId="y">
    <r:grantGroup>
      <r:grantGroup licensePartIdRef="y"/>
    </r:grantGroup>
  </r:grantGroup>
</r:license>
```

6.4 License issuers

Let l be an `r:License`. Let k be the number of `//r:issuer` children of l . If k is not 1, then the semantics of l is the same as the semantics of k independent `r:License` elements, one for each of the k `//r:issuer` children of l . Each of these k `r:License` elements λ is Equal to l , except that λ has only one `//r:issuer` child and that child is Equal to the respective `//r:issuer` child.

With the exception of signature verification, this processing step shall be carried out before the other License processing steps defined in this part of ISO/IEC 21000. In particular, it shall be carried out before `r:licensePartId` references, Variable references, and the testing of equality.

NOTE If k is 0, it follows that the semantics of l is the same as the semantics of no `r:License` at all. In colloquial terms, if a License has no issuer, it has not been issued and so has no effect.

6.5 Variables

6.5.1 Variable definition

6.5.1.1 General

A Variable is declared using an `r:forAll` element. A Variable has a name, a set of bindings, a scope, and a set of eligible bindings.

6.5.1.2 Variable name

Let f be an `r:forAll` element. Let y be the Variable declared by f . Then the name of y is the value of $f/@r:varName$.

6.5.1.3 Variable bindings

Let X be the universe of XML elements. Let q be an authorization request. Let e be an authorizer. Let f be an `r:forAll` element. Let y be the Variable declared by f . Then the set of bindings of y with respect to q and e is defined to be that subset of X such that x in X is in the set of bindings of y with respect to q and e if and only if a new XML document containing the element x as the root element Matches each of the $f/r:anXmlAttribute$ children of f with respect to q and e .

6.5.1.4 Variable scope

For any XML element x , let $\text{following}(x)$ be that set of XPath nodes selected by the XPath location path

```
following-sibling::*/*/*descendant-or-self::node()
```

when evaluated with x as the contextual XPath node. For an `r:forAll` element x , let $\text{redeclarations}(x)$ be that set of XPath nodes selected by the XPath location path

```
following-sibling::*/*/*descendant-or-self::r:forAll[@r:varName=$fVarName]
```

when evaluated with x as the contextual XPath node and $\$fVarName$ as the value of $x/@r:varName$. Let f be an `r:forAll` element. Let y be the Variable declared by f . Then the scope of y is defined to be $\text{following}(f)$ less the union of all $\text{following}(d)$ where each XPath node d is in $\text{redeclarations}(f)$.

6.5.1.5 Eligible Variable bindings

Let X be the universe of XML elements. Let q be an authorization request. Let e be an authorizer. Let f be an `r:forAll` element. Let y be the Variable declared by f . Then the set of eligible bindings of y with respect to q and e is defined to be that subset of X such that x in X is in the set of eligible bindings of y with respect to q and e if and only if x is in the set of bindings of y with respect to q and e and for all elements χ in the scope of y where $\chi/@r:varRef$ is Equal to $f/@r:varName$ all of the following hold:

- either the expanded element name of x exactly matches that of χ or x is substitutable for χ using substitution groups,
- either the type of x exactly matches that of χ or the type of x derives, through any number of levels, from the type of χ using type derivation, and
- if χ is removed from its document and replaced with an element Equal to x , that document is still valid.

6.5.2 Variable referencing

Let a be an `r:LicensePart`. If $a/@r:varRef$ exists, then a shall have empty content and the value of $a/@r:varRef$ shall be the name of some Variable whose scope includes a .

NOTE Conceptually, the contents of *a* are determined by the bindings of the Variable it references, so *a* doesn't need to have its own content.

6.6 Conceptually abstract elements and types

If a conceptually abstract element appears in an `r:License`, it shall either have a type that is not conceptually abstract or appear in the form of a Variable reference, as described in 6.5.2.

If a conceptually abstract type appears in an `r:License`, it shall belong to an element that either is not conceptually abstract or appears in the form of a Variable reference, as described in 6.5.2.

6.7 Revocable

A revocable is an ordered pair containing the following members, in order:

- a) a document and
- b) the `r:Principal` identifying the issuer of that document.

6.8 Principal surpassing

6.8.1 General

Let *a* and *b* be `r:Principal` elements. Let *A* be the set representation of *a* as defined in 6.8.2. Let *B* be the set representation of *b* as defined in 6.8.2. Then *b* Surpasses *a* if and only if for every α in *A* there exists a β in *B* such that α is Equal to β .

6.8.2 Principal set representation

Let *p* be an `r:Principal`. If *p* is an `r:AllPrincipals` element, then the set representation of *p* is the union of the set representations of each *p/r:principal* child of *p*. If *p* is not an `r:AllPrincipals` element, then the set representation of *p* is the one-member set containing only *p*.

6.9 Derived grants and grant groups

6.9.1 General

Let *a* be an `r:Grant` or an `r:GrantGroup`. Let *b* be an `r:Grant` or an `r:GrantGroup`. Let *q* be an authorization request. Let *e* be an authorizer. Then *a* is Derived from *b* with respect to *q* and *e* if and only if either of the following are true:

- *a* is one-step-derived from *b*, or
- there exists some `r:Grant` or `r:GrantGroup` ϕ such that all of the following are true:
 - *a* is one-step-derived from ϕ with respect to *q* and *e* as defined in 6.9.2.1, and
 - ϕ is Derived from *b* with respect to *q* and *e*.

6.9.2 One-step-derived grants and grant groups

6.9.2.1 General

Let *a* be an `r:Grant` or an `r:GrantGroup`. Let *b* be an `r:Grant` or an `r:GrantGroup`. Let *q* be an authorization request. Let *e* be an authorizer. Then *a* is one-step-derived from *b* with respect to *q* and *e* if and only if at least one of the following is true:

- *a* is Equal to *b*,

- a is one-step-derived from b with respect to q and e as defined in 6.9.2.2,
- a is one-step-derived from b with respect to q and e as defined in 6.9.2.3, or
- a is one-step-derived from b with respect to q and e as defined in 6.9.2.4.

6.9.2.2 One-step-derived grants and grant groups resulting from a Variable

Let a be an `r:Grant` or an `r:GrantGroup`. Let b be an `r:Grant` or an `r:GrantGroup`. Let q be an authorization request. Let e be an authorizer. Then a is one-step-derived from b with respect to q and e if and only if both $b/r:forall$ is not absent and, letting f be the first $b/r:forall$ child of b , a is Equal to b except that

- f is not present in a and
- there exists an x in the eligible bindings of the Variable declared by f with respect to q and e such that, throughout the scope of that Variable in b , all elements having references to that Variable are replaced in a by x .

6.9.2.3 One-step-derived grants and grant groups resulting from a grant group

Let a be an `r:Grant` or an `r:GrantGroup`. Let b be an `r:Grant` or an `r:GrantGroup`. Let q be an authorization request. Let e be an authorizer. Then a is one-step-derived from b with respect to q and e if and only if both b is an `r:GrantGroup` that has no child `r:forall` element and there exists a $b/r:grant$ or $b/r:grantGroup$ β such that a is Equal to β except that

- $a/r:principal$ is Equal to an `r:allPrincipals` whose children are, in order, $b/r:principal$ (if present) and $\beta/r:principal$ (if present) and
- $a/r:condition$ is Equal to an `r:allConditions` whose children are, in order, $b/r:condition$ (if present) and $\beta/r:condition$ (if present).

6.9.2.4 One-step-derived grants resulting from a delegation control

Let a be an `r:Grant` or an `r:GrantGroup`. Let b be an `r:Grant` or an `r:GrantGroup`. Let q be an authorization request. Let e be an authorizer. Then a is one-step-derived from b with respect to q and e if and only if all of the following are true:

- $b/r:delegationControl$ is present,
- for each $b/r:forall$ f , $b/r:delegationControl$, $b/r:principal$, and all of their descendent elements that are within the scope of the Variable declared by f do not have references (see 6.5.2) to the Variable declared by f ,
- a has no child `r:forall` elements,
- $a/r:delegationControl$ is absent,
- $a/r:principal$ is Equal to $b/r:principal$,
- $a/r:right$ is Equal to I ,
- letting α be $a/r:resource$, α is Equal to b except that
 - $\alpha/r:delegationControl$ is an allowable destination delegation control of $b/r:delegationControl$ with respect to q , e , and b as defined in 7.3.4.3,

- *a/r:principal* is an allowable destination principal of *b/r:delegationControl* with respect to *q*, *e*, and *b* as defined in 7.3.4.1, and
- *a/r:condition* is an allowable destination condition of *b/r:delegationControl* with respect to *q*, *e*, and *b* as defined in 7.3.4.2, and
- *a/r:condition* is absent.

NOTE By placing an *r:delegationControl* in an *r:Grant* or *r:GrantGroup*, the effect of delegation of that *r:Grant* or *r:GrantGroup* can be achieved. If Principal A issues a License to Principal B and enables delegation using an *r:delegationControl*, Principal B can then issue a similar License to Principal C who can then issue a similar License to Principal D, who might be the same as Principal B. If the Conditions in the License from Principal A to Principal B expire (they are no longer Satisfied), the Licenses to Principal C and Principal D are still valid. If this is not the intent of Principal A, Principal A can use appropriate Conditions and place appropriate *r:DcConstraint* elements in the *r:delegationControl* so that the Conditions applicable to Principal C and Principal D are at least as restrictive as (meaning they will expire no later than) those applicable to Principal B.

6.10 Encrypted Elements

In cases where an *r:License*, *r:Grant*, or *r:GrantGroup* has a child validating against the *r:encryptedLicense*, *r:encryptedGrant*, or *r:encryptedGrantGroup* particle, respectively, that child shall represent (see 7.3.6) the encryption of the contents of the *r:License*, *r:Grant*, or *r:GrantGroup*, and this part of ISO/IEC 21000 excepting 6.10 shall apply to the clear-text form of the *r:License*, *r:Grant*, or *r:GrantGroup*.

7 Core

7.1 General

This Clause specifies the syntax and semantics integral to the architecture.

7.2 Core authorization context properties

Table 3 specifies the authorization context properties relating to Clause 7 and the statements they represent. If a property has the name given in the first column of Table 3 and the value given in the second column of Table 3, then the statement represented by that property is the statement given in the third column of Table 3.

Table 3 — Core authorization context properties

Property name	Property value	Statement represented
<i>r:exerciseMechanism()</i>	<i>m</i>	<i>m</i> is an <i>r:ExerciseMechanism</i> , and the mechanism indicated by <i>m</i> is used to effect the requested performance.
<i>r:freshness(m, b, τ)</i>	true	<i>m</i> is an <i>r:RevocationMechanism</i> , <i>b</i> is a revocable, <i>τ</i> is a time instant, and the mechanism indicated by <i>m</i> for effecting revocation of a revocable guarantees that no revocation of <i>b</i> has been, is, or will be in effect before or at <i>τ</i> .
<i>r:fulfiller()</i>	<i>p</i>	<i>p</i> is an <i>r:Principal</i> , and <i>p</i> identifies the fulfiller (according to the semantics of the Right Member of the authorization request) for the requested performance.

7.3 General core elements and types

7.3.1 License

7.3.1.1 General

Additional normative semantics of `r:License` is given in 5.8, 6.3, 6.4, 6.6, and 6.10.

NOTE Example Licenses can be found in Annex D and G.5.

7.3.1.2 Title

For an `r:License` *l*, each of the `//r:title` elements that are present shall provide a descriptive phrase about *l* that is intended for human consumption.

EXAMPLE The content of an `r:title` element is displayed in a user interface.

Automated processors shall not interpret semantically the contents of such `//r:title` elements.

7.3.1.3 Inventory

An `r:Inventory` is a syntactic container of `r:LicensePart` elements. Automated processors shall not interpret semantically the contents of an `r:Inventory` element.

NOTE An `r:Inventory` element of an `r:License` provides a convenient place to define `r:LicensePart` elements without the definition site associating any particular semantics with the `r:LicensePart` elements. Usefully and usually, `r:LicensePart` elements in an `r:Inventory` will have `r:licensePartId` attributes so that they can be referenced elsewhere in the `r:License`.

7.3.1.4 OtherInfo

For an `r:License` *l*, the `//r:otherInfo` element, if present, shall provide additional information. Automated processors need not interpret semantically the contents of an `//r:otherInfo` element.

EXAMPLE A License creator might include in the `r:otherInfo` element some information that is peripherally related to some authentication or authorization process.

7.3.1.5 License attributes

For an `r:License` *l*, the `//@r:licenseId` attribute, if present, shall provide the URI that identifies *l*. Automated processors need not interpret semantically the contents of any `r:License` attributes.

7.3.1.6 Issuer

7.3.1.6.1 In a License

Let *l* be an `r:License` with one `//r:issuer` *u* (see 6.4 for additional semantics of `r:issuer` in Licenses). *u* shall give information about the issuer of *l*.

7.3.1.6.2 In general

Let *u* be an `r:Issuer`. `u/dsig:Signature`, if present, shall be a signature (XMLDSIG) that may be used to establish which Principal the issuer is.

NOTE If a License has been tampered with so that the digest value in the signature does not match the digest of the License, then the signature probably does not establish which Principal the issuer is, and therefore the tampered License does not follow the semantics of this subclause. In this way, signatures also serve as a way to ensure integrity and authenticity of Licenses.

u/r:principal, if present, shall identify which Principal the issuer is. *u/r:details*, if present, shall give details about the circumstances under which the issuer issues.

7.3.1.6.3 IssuerDetails

Let *e* be an *r:IssuerDetails*. *e/r:timeOfIssue*, if present, shall indicate the specific date and time at which the issuer claims to have issued. Each *e/r:revocationMechanism*, if any are present, shall indicate a mechanism by which the issuer shall, if he later wishes to revoke his revocable for the document whose issuer information is given by the parent of *e*, effect his revocation of that revocable.

EXAMPLE

```
<r:details>
  <r:timeOfIssue>2003-07-01T00:00:00</r:timeOfIssue>
  <r:revocationMechanism>
    <r:revocationService>
      <r:serviceReference licensePartIdRef="uddiService"/>
    </r:revocationService>
  </r:revocationMechanism>
</r:details>
```

7.3.1.6.4 RevocationMechanism

Let *m* be an *r:RevocationMechanism*. The child of *m* shall indicate a mechanism for effecting revocation of a revocable. *m/r:revocationService*, if present, indicates that the service reference identified by *m/r:revocationService/r:serviceReference* is the mechanism for effecting revocation of a revocable.

7.3.1.6.5 Signature recommendations

When a *dsig:Signature* *s* is used in an *r:Issuer* in an *r:License*, there should be an *s/dsig:SignedInfo/dsig:Reference* element *r* such that

- *r/@dsig:URI* is omitted and
- *r/dsig:Transforms* is present and contains exactly two child *dsig:Transform* elements, both *dsig:Transform* children are empty, the *dsig:Algorithm* attribute of the first *dsig:Transform* child has the value `urn:mpeg:mpeg21:2003:01-REL-R-NS:licenseTransform`, and the *dsig:Algorithm* attribute of the second *dsig:Transform* child has the value `urn:uddi-org:schemaCentricC14N:2002-07-10`.

NOTE 1 This practice simplifies the process of determining exactly which pieces of an *r:License* have actually been signed. It also ensures that signatures are consistent with the Equal operation.

NOTE 2 There can also be other *s/dsig:SignedInfo/dsig:Reference* elements that reference items (such as metadata) external to the License. Such references would be included in the signature. The use of such references is beyond the scope of this specification.

EXAMPLE

```
<dsig:Signature>
  <dsig:SignedInfo>
    <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <dsig:Reference>
      <dsig:Transforms>
        <dsig:Transform Algorithm="urn:mpeg:mpeg21:2003:01-REL-R-NS:licenseTransform"/>
        <dsig:Transform Algorithm="urn:uddi-org:schemaCentricC14N:2002-07-10"/>
      </dsig:Transforms>
      <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <dsig:DigestValue>Jk9QbKQCo941tTExbj1/Q==</dsig:DigestValue>
```

```

    </dsig:Reference>
  </dsig:SignedInfo>
  <dsig:SignatureValue>DFgqOhh5QQ==</dsig:SignatureValue>
  <dsig:KeyInfo>
    <dsig:KeyValue>
      <dsig:RSAKeyValue>
        <dsig:Modulus>gOyM4ccyzA==</dsig:Modulus>
        <dsig:Exponent>AQABAA==</dsig:Exponent>
      </dsig:RSAKeyValue>
    </dsig:KeyValue>
  </dsig:KeyInfo>
</dsig:Signature>

```

7.3.1.6.6 License transform

Let t be a `dsig:Transform` element with a `dsig:Algorithm` attribute whose value is `urn:mpeg:mpeg21:2003:01-REL-R-NS:licenseTransform`. Let l be the most immediate `r:License` ancestor of t . Then t emits as output l with all its `//r:issuer` children wholly removed except for that `//r:issuer` that contains t , which is kept, removing its `dsig:Signature` child instead.

7.3.1.7 Recommendation on declaration of Namespaces used in a License

An `r:License` should not rely on Namespace declarations to be imported from its context in its surrounding XML document.

NOTE Following this recommendation greatly facilitates the ability to manipulate an `r:License` as a self-contained unit within Applications. It also helps ensure that namespace declarations are included in signatures.

7.3.2 LicenseGroup

An `r:LicenseGroup` is a syntactic container of `r:License` elements. There is no normative semantics for `r:LicenseGroup`. No semantics is conveyed by the presence of two `r:License` elements within the same `r:LicenseGroup`.

NOTE This type exists merely due to the observation that it is often convenient to be able to use it as a container in XML documents. No use of it is made in this part of ISO/IEC 21000.

7.3.3 ForAll

The semantics of `r:forAll` is given in 6.5.

7.3.4 DelegationControl

7.3.4.1 Allowable destination principals

Let p be an `r:Principal`, d be an `r:delegationControl`, q be an authorization request, e be an authorizer, and g be an `r:Grant` or `r:GrantGroup`. Then p is an allowable destination principal of d with respect to q , e , and g if and only if p is an Allowable Destination Principal of each `d/r:dcConstraint` child of d with respect to q , e , and g .

NOTE Because an Allowable Destination Principal does not appear in a License, it and its descendents cannot have any references to Variables not declared within the Allowable Destination Principal itself.

7.3.4.2 Allowable destination conditions

Let c be an `r:Condition`, d be an `r:delegationControl`, q be an authorization request, e be an authorizer, and g be an `r:Grant` or `r:GrantGroup`. Then c is an allowable destination condition of d with respect to q , e , and g if and only if c is an Allowable Destination Condition of each `d/r:dcConstraint` child of d with respect to q , e , and g .

NOTE Because an Allowable Destination Condition does not appear in a License, it and its descendents cannot have any references to Variables not declared within the Allowable Destination Condition itself.

7.3.4.3 Allowable destination delegation controls

Let δ be an `r:delegationControl`, d be an `r:delegationControl`, q be an authorization request, e be an authorizer, and g be an `r:Grant` or `r:GrantGroup`. Then δ is an allowable destination delegation control of d with respect to q , e , and g if and only if δ is an Allowable Destination Delegation Control of each `dlr:dcConstraint` child of d with respect to q , e , and g .

NOTE Because an Allowable Destination Delegation Control does not appear in a License, it and its descendents cannot have any references to Variables not declared within the Allowable Destination Delegation Control itself.

7.3.5 NonSecureReference

The semantics and processing associated with the type `r:NonSecureReference` are identical to those of `dsig:ReferenceType`, except that the semantics and processing associated with the `dsig:DigestMethod` and `dsig:DigestValue` elements found in `dsig:ReferenceType` are omitted.

7.3.6 EncryptedContent

The type `r:EncryptedContent` modifies the semantics of its base type, `xenc:EncryptedDataType`. An `r:EncryptedContent` shall have an `xenc:Type` attribute with a value of `http://www.w3.org/2001/04/xmlenc#Content`. (See XMLENC.)

NOTE This value is the type associated with encrypting XML element content.

The plaintext of an `r:EncryptedContent` element shall replace semantically the `r:EncryptedContent` element and thus become the content of said element's parent. In doing so, the plaintext shall conform to the schema of the parent as a whole.

7.4 Core conceptually abstract elements and types

7.4.1 LicensePart

The element `r:licensePart` is conceptually abstract. The type `r:LicensePart` is conceptually abstract. Because of the requirements given in 6.3 c) and 6.5.2, if the syntax of a particular derivation of the type `r:LicensePart` declares its content model as optional, the semantics of that optional content model, unless specified otherwise, is that it shall not be omitted in an `r:LicensePart` of that derivation unless that `r:LicensePart` has an `r:licensePartIdRef` or `r:varRef` attribute.

7.4.2 Principal

The element `r:principal` is conceptually abstract. The type `r:Principal` is conceptually abstract. An `r:Principal` shall identify a system entity.

7.4.3 Right

The element `r:right` is conceptually abstract. The type `r:Right` is conceptually abstract. An `r:Right` shall identify an act. The semantic specification of each different particular kind of `r:Right` should indicate which kinds of `r:Resource`, if any, are to be used as the Resource Member of an authorization request having that kind of `r:Right` as the Right Member.

7.4.4 Resource

The element `r:resource` is conceptually abstract. The type `r:Resource` is conceptually abstract. An `r:Resource` shall identify an entity, quality, event, state, concept, substance, or anything else referred to by a noun.

7.4.5 Condition

The element `r:condition` is conceptually abstract. The type `r:Condition` is conceptually abstract. The semantics specification of each different particular kind of `r:Condition` shall indicate whether or not an `r:Condition` of that kind is Satisfied with respect to a given authorization request and authorization story. An application that encounters an `r:Condition` of which it lacks semantic knowledge shall not consider the `r:Condition` Satisfied.

7.4.6 AnXmlPatternAbstract

7.4.6.1 General

The element `r:anXmlPatternAbstract` is conceptually abstract. The type `r:AnXmlPatternAbstract` is conceptually abstract. The semantics specification of each different particular kind of `r:AnXmlPatternAbstract` shall indicate whether or not a given XML document Matches an `r:AnXmlPatternAbstract` of that kind with respect to a given authorization request and authorizer. An application that encounters an `r:AnXmlPatternAbstract` of which it lacks semantic knowledge shall not consider anything as Matching that `r:AnXmlPatternAbstract`.

7.4.6.2 PrincipalPatternAbstract

The element `r:principalPatternAbstract` is conceptually abstract. The type `r:PrincipalPatternAbstract` is conceptually abstract. An XML document shall not Match an `r:PrincipalPatternAbstract` with respect to any given authorization request and authorizer unless that XML document contains exactly one `r:Principal` as the root element.

7.4.6.3 RightPatternAbstract

The element `r:rightPatternAbstract` is conceptually abstract. The type `r:RightPatternAbstract` is conceptually abstract. An XML document shall not Match an `r:RightPatternAbstract` with respect to any given authorization request and authorizer unless that XML document contains exactly one `r:Right` as the root element.

7.4.6.4 ResourcePatternAbstract

The element `r:resourcePatternAbstract` is conceptually abstract. The type `r:ResourcePatternAbstract` is conceptually abstract. An XML document shall not Match an `r:ResourcePatternAbstract` with respect to any given authorization request and authorizer unless that XML document contains exactly one `r:Resource` as the root element.

7.4.6.5 ConditionPatternAbstract

The element `r:conditionPatternAbstract` is conceptually abstract. The type `r:ConditionPatternAbstract` is conceptually abstract. An XML document shall not Match an `r:ConditionPatternAbstract` with respect to any given authorization request and authorizer unless that XML document contains exactly one `r:Condition` as the root element.

7.4.7 DcConstraint

The element `r:dcConstraint` is conceptually abstract. The type `r:DcConstraint` is conceptually abstract.

The semantics specification of each different particular kind of `r:DcConstraint` shall indicate whether or not a given `r:Principal` is an Allowable Destination Principal of an `r:DcConstraint` of that kind with respect to a given authorization request, authorizer, and `r:Grant` or `r:GrantGroup`. An application that encounters an `r:DcConstraint` of which it lacks semantic knowledge shall not consider anything as an Allowable Destination Principal of that `r:DcConstraint`.

The semantics specification of each different particular kind of `r:DcConstraint` shall indicate whether or not a given `r:Condition` is an Allowable Destination Condition of an `r:DcConstraint` of that kind with respect to a given authorization request, authorizer, and `r:Grant` or `r:GrantGroup`. An application that encounters an `r:DcConstraint` of which it lacks semantic knowledge shall not consider anything as an Allowable Destination Condition of that `r:DcConstraint`.

The semantics specification of each different particular kind of `r:DcConstraint` shall indicate whether or not a given `r:delegationControl` is an Allowable Destination Delegation Control of an `r:DcConstraint` of that kind with respect to a given authorization request, authorizer, and `r:Grant` or `r:GrantGroup`. An application that encounters an `r:DcConstraint` of which it lacks semantic knowledge shall not consider anything as an Allowable Destination Delegation Control of that `r:DcConstraint`.

7.4.8 TrustRoot

The element `r:trustRoot` is conceptually abstract. The type `r:TrustRoot` is conceptually abstract. The semantics specification of each different particular kind of `r:TrustRoot` shall indicate the set of `r:Grant` elements identified by an `r:TrustRoot` of that kind with respect to a given authorization request and authorizer. An application that encounters an `r:TrustRoot` of which it lacks semantic knowledge shall consider that `r:TrustRoot` as representing the empty set.

7.4.9 ServiceDescription

The element `r:serviceDescription` is conceptually abstract. The type `r:ServiceDescription` is conceptually abstract. An `r:ServiceDescription` shall describe a service. An `r:ServiceDescription` should indicate the address of, the interface to, the rules for interacting with, and the functionality provided by the service. An `r:ServiceDescription` may include additional information about a service.

NOTE See 8.12 for some examples of concrete types derived from `r:ServiceDescription`.

7.4.10 PropertyAbstract

The element `r:propertyAbstract` is conceptually abstract. The type `r:PropertyAbstract` is conceptually abstract. An `r:PropertyAbstract` shall identify a property that can be possessed by a Principal.

7.5 Core principals

7.5.1 AllPrincipals

Let p be an `r:AllPrincipals`. Then p identifies that system entity that is identified by each of the $p/r:principal$ children of p .

EXAMPLE

```
<r:allPrincipals>
  <r:keyHolder licensePartIdRef="Alice"/>
  <r:keyHolder licensePartIdRef="Alice2"/>
</r:allPrincipals>
```

NOTE If p has no children, then, according to 6.8, any `r:Principal` Surpasses p .

7.5.2 KeyHolder

Let p be an `r:KeyHolder`. Then p identifies that system entity that possesses the cryptographic key indicated by the $p/r:info$ element as determined by the semantics of `dsig:KeyInfoType` given in XMLDSIG.

EXAMPLE 1 A Principal using public-key cryptography can be identified as that Principal that possesses the private key that corresponds to the public key given in an `r:KeyHolder`.

EXAMPLE 2

```

<r:keyHolder licensePartId="Alice">
  <r:info>
    <dsig:KeyValue>
      <dsig:RSAKeyValue>
        <dsig:Modulus>AliM4ccyzA==</dsig:Modulus>
        <dsig:Exponent>AQABAA==</dsig:Exponent>
      </dsig:RSAKeyValue>
    </dsig:KeyValue>
  </r:info>
</r:keyHolder>

```

7.6 Core rights

7.6.1 Issue

Let r be an $r:issue$. Then r identifies the act of including an $r:Grant$ or $r:GrantGroup$ as $//r:grant$ or $//r:grantGroup$ when issuing an $r:License$ l .

If r is used as the Right Member of an authorization request, then the Resource Member of that authorization request shall be present and shall be an $r:Grant$ or an $r:GrantGroup$.

NOTE Several authorization requests apply to issuing an $r:License$ l , one for each $//r:grant$ or $//r:grantGroup$ therein.

7.6.2 Obtain

Let r be an $r:Obtain$. Then r identifies the act of obtaining an $r:Grant$ or $r:GrantGroup$ as $//r:grant$ or $//r:grantGroup$ in an issued $r:License$ l .

If r is used as the Right Member of an authorization request, then the Resource Member of that authorization request shall be present and shall be an $r:Grant$ or an $r:GrantGroup$ and, letting Σ be the Authorization Context Member of that authorization request, $\Sigma.r:fulfiller()$ shall identify the Principal who is requested to be the issuer of l .

NOTE 1 The use of an $r:Obtain$ can be conceptualized as an offer or advertisement possibly for the sale of some $r:Grant$ or $r:GrantGroup$. Suppose there is an authorization proof for an authorization request having an $r:Obtain$ as the Right Member, an accurate Authorization Context Member Σ , and a Trust Root Member R . If R represents a sole trusted root issuer that is the same Principal identified by $\Sigma.r:fulfiller()$, it is generally expected that that Principal indeed will fulfill the request to obtain, since he is ultimately responsible for permitting the act of obtaining.

NOTE 2 If a Principal is permitted to obtain an $r:Grant$ from another Principal, there is no implication that the other Principal is permitted to issue that $r:Grant$.

NOTE 3 If a Principal is permitted to obtain an $r:Grant$, there is no implication that the Principal is permitted to do the things mentioned in that $r:Grant$ until he actually obtains it, at which time the normal License semantics apply to the License that has the obtained $r:Grant$.

EXAMPLE

```

<r:grant>
  <r:keyHolder licensePartIdRef="Alice"/>
  <r:obtain/>
  <r:grantGroup licensePartIdRef="playAndAdapt"/>
</r:grant>

```

7.6.3 PossessProperty

Let r be an $r:PossessProperty$. Then r identifies the act of claiming ownership of a property.

If r is used as the Right Member of an authorization request, then the Resource Member of that authorization request shall be present and shall be an `r:PropertyAbstract`.

NOTE An `r:PossessProperty` can be used to model the authorized bindings of names and other attributes to Principals as is done in X.509 (ISO/IEC 9594-8) and other public key certificates. An `r:PossessProperty` can also be used to model group membership and roles.

7.6.4 Revoke

Let r be an `r:Revoke`. Then r identifies the act of utilizing a mechanism to effect the revocation of a revocable.

If r is used as the Right Member of an authorization request, then the Resource Member of that authorization request shall be present and shall identify a revocable.

NOTE 1 Issuing a License having an `r:Grant` having an `r:Revoke` does not effect any revocation. Rather, one typically issues a License having an `r:Grant` having an `r:Revoke` in order to allow other Principals to effect the revocation of a revocable.

NOTE 2 Suppose one License allows a user to do something. Suppose a second License allows a second user to revoke a revocable for the first License. Typically the first user will not be involved with the second License. Rather, there will be a revocation mechanism indicated in the first license, and that revocation mechanism will rely on the second License to determine if (within the trust roots of that revocation mechanism) the second user's revoke is permitted. Then, the first user will rely on that revocation mechanism to determine whether the first License is still in force.

NOTE 3 It is expected that many revocation mechanisms will have trust roots installed that allow an issuer (and only the issuer) of a License to revoke the revocable for his License and freely delegate that ability to others.

NOTE 4 Because a revocation mechanism does not revoke but simply records the revocation, users do not check if the revocation mechanism is permitted to revoke.

EXAMPLE

```
<r:grant>
  <r:keyHolder licensePartIdRef="Alice"/>
  <r:revoke/>
  <r:revocable licensePartIdRef="licenseToBeRevoked"/>
</r:grant>
```

7.7 Core resources

7.7.1 DigitalResource

Let t be an `r:DigitalResource`. Then t identifies an arbitrary sequence of digital bits b , determined as follows:

- If `t/r:anXml` is present, then b is the character string that is the sequence of zero or more XML elements contained within `t/r:anXml`.
- If `t/r:binary` is present, then b is the bit sequence whose base64 encoding is the value of `t/r:binary`.
- If `t/r:secureIndirect` is present, then b is the bit sequence to which that element refers according to the semantics and processing associated with the type `dsig:ReferenceType` (XMLDSIG).
- If `t/r:nonSecureIndirect` is present, then b is the bit sequence to which that element refers according to the semantics and processing associated with the type `r:NonSecureReference`.
- Otherwise, b is the bit sequence identified by the child of t , according to the semantics of that child.

NOTE When using `r:nonSecureIndirect`, the user should be comfortable with the fact that different executions of the process for determining the sequence of bits might yield different results. No cryptographic measures are in place to ensure the yielded sequence of bits is the expected one.

EXAMPLE

```
<r:digitalResource licensePartId="video">
  <r:nonSecureIndirect URI="http://acme.org/myVideo"/>
</r:digitalResource>
```

7.7.2 Grant

Let t be an `r:Grant`. Then t identifies the Resource that is an XML element Equal to t .

NOTE Additional semantics related to `r:Grant` elements is given in 5.7, 6.9, and 6.10.

7.7.3 GrantGroup

Let t be an `r:GrantGroup`. Then t identifies the Resource that is an XML element Equal to t .

NOTE Additional semantics related to `r:GrantGroup` elements is given in 5.7, 6.9, and 6.10.

EXAMPLE

```
<r:grantGroup licensePartId="playAndAdapt">
  <r:keyHolder licensePartIdRef="Alice"/>
  <r:grant>
    <mx:play/>
    <r:digitalResource licensePartIdRef="video"/>
  </r:grant>
  <r:grant>
    <mx:adapt/>
    <r:digitalResource licensePartIdRef="video"/>
  </r:grant>
</r:grantGroup>
```

7.7.4 Revocable

Let t be an `r:Revocable`. Then t identifies a revocable (l, p) , determined as follows:

- If `t/dsig:SignatureValue` is present, then, letting s be the `dsig:SignatureType` having a child Equal to `t/dsig:SignatureValue`, both l is the document to which s applies and p is the `r:Principal` identifying that issuer of l that is established by s .
- If `t/dsig:Reference` is present, then both of the following are true:
 - `t/dsig:Reference` shall reference a `dsig:SignatureType` and,
 - letting s be that `dsig:SignatureType`, both l is the document to which s applies and p is the `r:Principal` identifying that issuer of l that is established by s .
- If `t/r:licenseId` is present, then both p is Equal to `t/r:principal` and l is the `r:License` where all of the following are true:
 - the value of `//@r:licenseId` is the value of `t/r:licenseId`,
 - there is one `//r:issuer`, and
 - `//r:issuer` corresponds to p .

EXAMPLE

```
<r:revocable licensePartId="licenseToBeRevoked">
  <r:licenseId>urn:acme:license:id:12345</r:licenseId>
  <r:keyHolder licensePartIdRef="acme"/>
</r:revocable>
```

7.7.5 ServiceReference

Let t be an `r:ServiceReference`. Then t identifies a reference to a service. The service referred to by t is described by `t/r:serviceDescription`. If `t/r:serviceParameters` is present, the values of the reference-specific parameters needed to interact with the service according to `t/r:serviceDescription` are determined according to the following paragraph, letting p be `t/r:serviceParameters`. If `t/r:serviceParameters` is not present, no reference-specific parameters are needed to interact with the service.

For each integer k between 1 and the number of `p/r:datum` elements, let d_k be the k^{th} `p/r:datum` element. If there are no elements following d_k or if the element immediately following d_k is not an `r:transforms`, then the value of the k^{th} reference-specific parameter needed to interact with the service according to `t/r:serviceDescription` is the child of d_k . If there is an element immediately following d_k and that element is an `r:transforms`, then the value of the k^{th} reference-specific parameter needed to interact with the service according to `t/r:serviceDescription` is determined by the semantics of that `r:transforms` as given in XMLDSIG for `dsig:TransformType` and modified as follows:

- the input to the first `dsig:Transform` is an XPath node-set containing the child of d_k in the context of a new XML document containing that child as the root element, and
- the output of the last `dsig:Transform` is the value of the k^{th} reference-specific parameter needed to interact with the service according to `t/r:serviceDescription`.

NOTE The interpretation, detailed processing, and passing to the service of the reference-specific parameters is necessarily service-specific and is thus not defined here.

EXAMPLE 1

```
<r:serviceReference licensePartId="uddiService">
  <sx:uddi>
    <sx:serviceKey>
      <sx:uuid>D04951E4-332C-4693-B7DB-D3D1D1C21111</sx:uuid>
    </sx:serviceKey>
  </sx:uddi>
</r:serviceReference>
```

EXAMPLE 2

```
<r:serviceReference licensePartId="anonymousService">
  <sx:anonymousStateService/>
  <r:serviceParameters>
    <r:datum>
      <sx:stateDistinguisher>5001EB7E-80BF-43f7-9065-
7E98CE52279E</sx:stateDistinguisher>
    </r:datum>
  </r:serviceParameters>
</r:serviceReference>
```

7.8 Core conditions

7.8.1 AllConditions

Let c be an `r:AllConditions`. Let q be an authorization request. Let s be an authorization story. Then c is Satisfied with respect to q and s if and only if each `c/r:condition` child of c is Satisfied with respect to q and s .

EXAMPLE

```
<r:allConditions>
  <r:validityInterval licensePartIdRef="firstMonth"/>
  <r:exerciseMechanism>
    <r:exerciseService>
      <r:serviceReference licensePartIdRef="uddiService"/>
    </r:exerciseService>
  </r:exerciseMechanism>
</r:allConditions>
```

```

    </r:exerciseMechanism>
  </r:allConditions>

```

NOTE If c has no children, then c is Satisfied.

7.8.2 ExerciseMechanism

Let c be an $r:ExerciseMechanism$. Let $(p, r, t, v, \Sigma, L, R)$ be an authorization request. Let (g, h, e) be an authorization story. Then c is Satisfied with respect to $(p, r, t, v, \Sigma, L, R)$ and (g, h, e) if and only if c is Equal to $\Sigma.r:exerciseMechanism()$.

The child of c shall indicate a mechanism for effecting a performance. $c/r:exerciseService$, if present, indicates that the service reference indicated by $c/r:exerciseService/r:serviceReference$ is the mechanism for effecting a performance.

NOTE An $r:ExerciseMechanism$ is particularly useful when used in conjunction with an $r:Obtain$ (see Example 1).

EXAMPLE 1 In a super distribution scenario, it is common for the users who wish to obtain an $r:Grant$ to be very removed from the original channels of distribution. An $r:ExerciseMechanism$ could direct the user back to an official distribution channel.

EXAMPLE 2 A clerk may be permitted to insert records into a database only if he uses a particular (error-checking) user interface form designed for that purpose.

EXAMPLE 3 An airline company may permit its frequent flyers to ticket for a reduced fare when ticketing via a particular online travel service.

EXAMPLE 4

```

<r:exerciseMechanism>
  <r:exerciseService>
    <r:serviceReference licensePartIdRef="uddiService"/>
  </r:exerciseService>
</r:exerciseMechanism>

```

7.8.3 ExistsRight

Let c be an $r:ExistsRight$. Let $(p, r, t, v, \Sigma, L, R)$ be an authorization request. Let (g, h, e) be an authorization story. Let η be the $c/r:grant$ or $c/r:grantGroup$ child of c , whichever is present. Let $\emptyset$ be

- the set of $r:Grant$ elements identified by $c/r:trustRoot$ with respect to $(p, r, t, v, \Sigma, L, R)$ and authorizer e , if $c/r:trustRoot$ is present, or
- R , if $c/r:trustRoot$ is not present.

Then c is Satisfied with respect to $(p, r, t, v, \Sigma, L, R)$ and (g, h, e) if and only if there is an authorizer ε such that both of the following are true:

- if ε is absent, then η is a member of $\emptyset$, and
- if ε is present, then, letting (l, π, i, σ, s) be the ordered five-tuple representation of ε , all of the following are true:
 - l is a member of L ,
 - η is Equal to one of the $//r:Grant$ or $//r:GrantGroup$ children of l ,
 - $\Sigma.r:issueTime(l, \pi)$ is i ,
 - $\Sigma.r:issueContext(l, \pi, \eta, \sigma)$ is true,

- the time instant i is no later in time than the start of time interval v , and
- s is an authorization proof for the authorization request $(\pi, I, \eta, \psi, \sigma, L, \Theta)$ where ψ is a time interval of zero length starting at i .

EXAMPLE

```
<r:existsRight>
  <r:grant>
    <r:keyHolder licensePartIdRef="Alice"/>
    <mx:play/>
    <r:digitalResource licensePartIdRef="video"/>
    <r:validityInterval licensePartIdRef="firstMonth"/>
  </r:grant>
</r:existsRight>
```

7.8.4 Fulfiller

Let c be an $r:Fulfiller$. Let $(p, r, t, v, \Sigma, L, R)$ be an authorization request. Let (g, h, e) be an authorization story. Then c is Satisfied with respect to $(p, r, t, v, \Sigma, L, R)$ and (g, h, e) if and only if $c/r:principal$ is Equal to $\Sigma.r:fulfiller()$.

EXAMPLE

```
<r:fulfiller>
  <r:keyHolder licensePartIdRef="Alice"/>
</r:fulfiller>
```

7.8.5 PrerequisiteRight

Let c be an $r:PrerequisiteRight$. Let $(p, r, t, v, \Sigma, L, R)$ be an authorization request. Let (g, h, e) be an authorization story. Let Θ be

- the set of $r:Grant$ elements identified by $c/r:trustRoot$ with respect to $(p, r, t, v, \Sigma, L, R)$ and authorizer e , if $c/r:trustRoot$ is present, or
- R , if $c/r:trustRoot$ is not present.

Then c is Satisfied with respect to $(p, r, t, v, \Sigma, L, R)$ and (g, h, e) if and only if there is an authorization proof for the authorization request $(c/r:principal, c/r:right, c/r:resource, v, \Sigma, L, \Theta)$.

EXAMPLE

```
<r:prerequisiteRight>
  <r:keyHolder licensePartIdRef="Alice"/>
  <mx:play/>
  <r:digitalResource licensePartIdRef="video"/>
</r:prerequisiteRight>
```

7.8.6 RevocationFreshness

Let c be an $r:RevocationFreshness$. Let $(p, r, t, v, \Sigma, L, R)$ be an authorization request. Let (g, h, e) be an authorization story. Then c is Satisfied with respect to $(p, r, t, v, \Sigma, L, R)$ and (g, h, e) if and only if one of the following is true:

- e is absent, or
- both e is present and, letting (l, π, i, σ, s) be the ordered five-tuple representation of e and letting b be the revocable (l, π) , there exists an $r:RevocationMechanism$ m and a time instant τ such that all of the following are true:
 - m is Equal to one of the $//r:issuer/r:details/r:revocationMechanism$ elements,