

国际电信联盟

ITU-T

国际电信联盟
电信标准化部门

X.1142

(06/2006)

X系列：数据网、开放系统通信和安全
电信安全

可扩展的接入控制标记语言 (XACML 2.0)

ITU-T X.1142建议书

ITU-T



国际电信联盟

ITU-T X系列建议书
数据网、开放系统通信和安全

公众数据网	
业务和设施	X.1-X.19
接口	X.20-X.49
传输、信令和交换	X.50-X.89
网络概貌	X.90-X.149
维护	X.150-X.179
管理安排	X.180-X.199
开放系统互连	
模型和记法	X.200-X.209
服务限定	X.210-X.219
连接式协议规范	X.220-X.229
无连接式协议规范	X.230-X.239
PICS书写形式	X.240-X.259
协议标识	X.260-X.269
安全协议	X.270-X.279
层管理对象	X.280-X.289
一致性测试	X.290-X.299
网间互通	
概述	X.300-X.349
卫星数据传输系统	X.350-X.369
以IP为基础的网络	X.370-X.379
报文处理系统	X.400-X.499
号码簿	X.500-X.599
OSI 组网和系统概貌	
组网	X.600-X.629
效率	X.630-X.639
业务质量	X.640-X.649
命名、寻址和登记	X.650-X.679
抽象句法记法1(ASN.1)	X.680-X.699
OSI 管理	
系统管理框架和结构	X.700-X.709
管理通信服务和协议	X.710-X.719
管理信息的结构	X.720-X.729
管理功能ODMA功能	X.730-X.799
安全	X.800-X.849
OSI 应用	
托付、并发和恢复	X.850-X.859
事务处理	X.860-X.879
远程操作	X.880-X.889
ASN.1的一般应用	X.890-X.899
开放分布式处理	X.900-X.999
电信安全	X.1000-

可扩展的接入控制标记语言 (XACML 2.0)

摘 要

XACML 是用于表示接入控制策略的 XML 词汇。接入控制由决定是否允许请求接入资源以及执行该决定组成。本建议书规定核心 XACML，包括策略语言模型、句法和以及处理规则的语言、模型和上下文的句法。本建议书规定 XACML 基于核心和层次角色的接入控制轮廓。规定 XACML 的多资源轮廓以及 XACML 的 SAML 2.0 轮廓。为改进基于策略互换 XACML 的安全，本建议书也规定用于数据安全的 XAML XML 数字签名轮廓。为向实施提供指南，规定了专用轮廓。

本建议书在技术上等同于 OASIS XACML 2.0 标准，并与该标准相兼容。

来 源

ITU-T 第 17 研究组(2005-2008)按照 ITU-T A.8 建议书规定的程序，于 2006 年 6 月 13 日批准了 ITU-T X.1142 建议书。

前 言

国际电信联盟（ITU）是从事电信领域工作的联合国专门机构。ITU-T（国际电信联盟电信标准化部门）是国际电信联盟的常设机构，负责研究技术、操作和资费问题，并且为在世界范围内实现电信标准化，发表有关上述研究项目的建议书。

每四年一届的世界电信标准化全会（WTSA）确定 ITU-T 各研究组的研究课题，再由各研究组制定有关这些课题的建议书。

WTSA 第 1 号决议规定了批准建议书须遵循的程序。

属 ITU-T 研究范围的某些信息技术领域的必要标准，是与国际标准化组织（ISO）和国际电工技术委员会（IEC）合作制定的。

注

本建议书为简明扼要起见而使用的“主管部门”一词，既指电信主管部门，又指经认可的运营机构。

遵守本建议书的规定是以自愿为基础的，但建议书可能包含某些强制性条款（以确保例如互操作性或适用性等），只有满足所有强制性条款的规定，才能达到遵守建议书的目的。“应该”或“必须”等其它一些强制性用语及其否定形式被用于表达特定要求。使用此类用语不表示要求任何一方遵守本建议书。

知识产权

国际电联提请注意：本建议书的应用或实施可能涉及使用已申报的知识产权。国际电联对无论是其成员还是建议书制定程序之外的其它机构提出的有关已申报的知识产权的证据、有效性或适用性不表示意见。

至本建议书批准之日止，国际电联尚未收到实施本建议书可能需要的受专利保护的知识产权的通知。但需要提醒实施者注意的是，这可能并非最新信息，因此特大力提倡他们通过下列网址查询电信标准化局（TSB）的专利数据库：<http://www.itu.int/ITU-T/ipr/>。

© 国际电联 2007

版权所有。未经国际电联事先书面许可，不得以任何手段复制本出版物的任何部分。

目 录

	页码
1 范围.....	1
2 参考文献.....	1
3 定义.....	2
3.1 引入的定义.....	2
3.2 附加定义.....	3
4 缩写.....	5
5 惯例.....	5
6 概述.....	6
7 XACML 核心.....	6
7.1 背景.....	6
7.2 XACML 模型.....	10
7.3 XACML 上下文.....	11
7.4 策略句法.....	14
7.5 上下文句法.....	34
7.6 XACML 功能需求.....	41
7.7 XACML 扩展点.....	49
7.8 一致性.....	50
8 基于核心和层次角色的接入控制 (RBAC) 轮廓.....	57
8.1 RBAC 背景.....	57
8.2 RBAC 例子.....	59
8.3 指派和激活角色属性.....	63
8.4 实现 RBAC 的模型.....	65
8.5 轮廓.....	67
8.6 标识符.....	67
9 XACML 的多资源轮廓.....	68
9.1 请求多个资源.....	69
9.2 请求整个层次.....	71
9.3 新属性标识符.....	72
9.4 新轮廓标识符.....	73
10 XACML 的 SAML 2.0 轮廓.....	73
10.1 映射 SAML 和 XACML 属性.....	75
10.2 授权决定.....	76
10.3 策略.....	77
10.4 元素<saml:Assertion>.....	79
10.5 元素<samlp:RequestAbstractType>.....	80
10.6 元素<samlp:Response>.....	80
11 XML 数字签名轮廓.....	81
11.1 SAML 的使用.....	81
11.2 规范化.....	81
11.3 签名模式.....	82
12 XACML 的层次资源轮廓.....	82
12.1 表示节点的身份信息.....	83
12.2 请求对节点的接入.....	84
12.3 确定节点所适用的策略.....	87
12.4 新 DataType: xpath-expression.....	87
12.5 新属性标识符.....	88
12.6 新的轮廓标识符.....	88
13 保密策略轮廓.....	89
13.1 标准属性.....	89
13.2 标准规则: 匹配目的.....	89

	页码
附件 A — 数据类型和函数	90
A.1 引言	90
A.2 数据类型	90
A.3 函数	92
附件 B — XACML 标识符	104
B.1 XACML 命名空间	104
B.2 接入主体类别	104
B.3 数据类型	104
B.4 主体属性	105
B.5 资源属性	106
B.6 动作属性	106
B.7 环境属性	106
B.8 状态码	107
B.9 组合算法	107
附件 C — 组合算法	108
C.1 Deny-overrides	108
C.2 Ordered-deny-overrides	109
C.3 Permit-overrides	109
C.4 Ordered-Permit-overrides	111
C.5 First-applicable	111
C.6 Only-one-applicable	113
附件 D — XACML 模式定义	114
D.1 XACML 上下文模式定义	114
D.2 策略模式定义	116
D.3 XACML SAML 协议模式定义	122
D.4 XACML SAML 声明模式定义	123
附录一 — 安全考虑	124
I.1 威胁模型	124
I.2 安全措施	126
附录二 — XACML 示例	128
II.1 示例一	128
II.2 示例二	131
附录三 — 高阶包函数示例描述	145
III.1 高阶包函数示例	145
参考资料	150

可扩展的接入控制标记语言 (XACML 2.0)

1 范围

本建议书规定可扩展的接入控制标记语言 (XACML) 第 2 版。它规定表示安全策略的公用语言。XACML 的动机是开发基于 XML 的策略语言，用于：

- 提供用于将单独规则和策略组合到应用于特定决定请求的单一策略集的方法。
- 提供用于组合规则和策略规程的灵活定义的方法。
- 提供用于处理在不同能力下进行操作的多主体的方法。
- 提供用于根据主体和资源的属性进行授权决定的方法。
- 提供处理多值属性的方法。
- 提供根据信息资源的内容进行授权决定的方法。
- 提供有关主体、资源和环境属性的逻辑和数学操作符集。
- 提供用于在为提炼定位、检索和认证策略构件方法时，处理分布的策略构件集的方法。
- 根据主体、资源和动作的属性值，提供快速标识应用于给定动作的策略的方法。
- 提供将策略书写者与应用环境的细节相隔离的抽象层。
- 提供规定必须与策略执行一同执行的动作集的一种方法。

本建议书包括用于这些需求中每一个需求的 XACML 方案。特别地，第 7 节规定包括 XACML 模型、策略语言模型、策略句法以及处理规程的核心 XACML 语言。第 8 节规定 XACML 基于核心和层次角色的接入控制(RBAC)轮廓。第 9 节规定 XACML 多资源轮廓。第 10 节讨论通过开发用于 XACML 的 SAML 2.0 轮廓保证 XACML 通信安全的技术。第 11 节通过开发用于 XACML 的 XML 数字签名轮廓而构建第 10 节的基础。第 12 节讨论通过开发层次的 XACML 的资源轮廓根据在第 7 节到第 11 节中规定的 XACML 轮廓的组合。在第 13 节中讨论私有问题。

2 参考文献

下列建议书和其他参考文献的条款，通过在本建议书中的引用而构成本建议书的条款。在出版时，所指出的版本是有效的。所有的建议书和其他参考文献都会被修订，使用本建议书的各方应探讨使用下列建议书或其他参考文献最新版本的可能性。国际电联电信标准化局保留有当前有效的 ITU-T 建议书。IETF 保留 RFC 列表以及已经被后来的 RFC 替代的 RFC。W3C 保留最新的建议书及其他出版物的目录。

- ITU-T Recommendation X.811 (1995) | ISO/IEC 10181-2:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Authentication framework.*
- ITU-T Recommendation X.812 (1995) | ISO/IEC 10181-3:1996, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Access control framework.*
- ITU-T Recommendation X.1141 (2006), *Security Assertion Markup Language (SAML 2.0).*
- IETF RFC 822 (1982), *Standard for the Format of ARPA Internet Text Messages.*
- IETF RFC 2119 (1997), *Key words for use in RFCs to Indicate Requirement Levels.*
- IETF RFC 2253 (1997), *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names.*
- IETF RFC 2256 (1997), *A Summary of the X.500 (96) User Schema for use with LDAPv3.*

- IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax*.
- IETF RFC 2732 (1999), *Format for Literal IPv6 Addresses in URL's*.
- IETF RFC 2821 (2001), *Simple Mail Transfer Protocol*.
- IETF RFC 3280 (2002), *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.
- W3C Canonicalization:2002, *Exclusive XML Canonicalization Version 1.0*, W3C Recommendation, Copyright © [18 July 2002] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/xml-exc-c14n/>.
- W3C Datatypes:2001, *XML Schema Part 2: Datatypes*, W3C Recommendation, Copyright © [2 May 2001] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>.
- W3C Encryption:2002, *XML Encryption Syntax and Processing*, W3C Recommendation, Copyright © [10 December 2002] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>.
- W3C MathML:2003, *Mathematical Markup Language (MathML), Version 2.0*, W3C Recommendation, Copyright © [21 October 2003] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2003/REC-MathML2-20031021/>.
- W3C Signature:2002, *XML-Signature Syntax and Processing*, W3C Recommendation, Copyright © [12 February 2002] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/2002/REC-xmlsig-core-20020212/>.
- W3C XML:2004, *Extensible Markup Language (XML) 1.0 (Third Edition)*, W3C Recommendation, Copyright © [4 February 2004] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/REC-xml/>.
- W3C XPATH:1999, *XML Path Language, Version 1.0*, W3C Recommendation, Copyright © [16 November 1999] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/1999/REC-xpath-19991116/>.
- W3C XSLT:1999, *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation, Copyright © [16 November 1999] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), <http://www.w3.org/TR/1999/REC-xslt-19991116/>.

注 — 本建议书引用的文件自成一体时不具备建议书的地位。

3 定义

就本建议书而言，下列定义适用。

3.1 引入的定义

3.1.1 本建议书采用 ITU-T X.811 建议书中规定的下列术语：

- a) 原则。

3.1.2 本建议书采用 ITU-T X.812 建议书中规定的下列术语：

- a) 接入控制信息；
- b) 用户。

3.1.3 本建议书采用 W3C Web 业务术语表中规定的下列术语：

- a) 命名空间；
- b) XML 方案。

3.1.4 本建议书采用 IETF RFC 2828 中规定的下列术语:

- a) 接入;
- b) 接入控制;
- c) 策略管理点;
- d) 策略决定点;
- e) 策略执行点;
- f) 安全体系架构;
- g) 安全策略;
- h) 安全服务。

3.1.5 本建议书采用 IETF 2396 中规定的下列术语

- a) 统一资源标识符 (URI) ;
- b) URI参考。

3.1.6 本建议书采用 W3C XML 签名中规定的下列术语:

- a) 数据客体。

3.2 附加定义

3.2.1 access 接入: 执行一个动作。

3.2.2 access control 接入控制: 根据策略控制接入。

3.2.3 action 动作: 针对资源的一个操作。

3.2.4 applicable policy 可应用的策略: 对特定决定请求管理接入的策略以及策略集的集合。

3.2.5 attribute 属性: 在一个声明或目标中可以引用的主体、动作或环境的特征。

3.2.6 attribute authority (AA) 属性机构 (AA): 将属性绑定到标识符的实体。这样的绑定可以使用具有作为签发者的属性机构的 SAML 属性声明表示。

3.2.7 authorization decision 授权决定: 赋值由 PDP 返回到 PEP 的可应用的策略的结果。赋值到“Permit”、“Deny”、“Indeterminate”或“NotApplicable”的功能以及职责集(可选)。

3.2.8 bag 包: 值的无序组合, 在其中可能有重复的值。

3.2.9 condition 条件: 判断的表达式。赋值“True”、“False”或“Indeterminate”的功能。

3.2.10 conjunctive sequence 连接次序: 使用逻辑的“与”操作组合的断言次序。

3.2.11 context 上下文: 决定请求和授权决定的标准表示。

3.2.12 context handler 上下文处理器: 将采用自身请求格式表示的决定请求转换为 XACML 标准格式并将以 XACML 标准格式表示的授权决定转换为自身响应格式的系统实体。

3.2.13 custodian 管理人: 个性化可确认的信息被委托的实体。

3.2.14 data object 数据客体: 指被签署的数字客体。使用 URI 在一个<reference>单元中被引用的数据客体。

3.2.15 decision 决定: 赋值一个规则、策略或策略集的结果。

3.2.16 decision request 决定请求: 为实施一个授权的决定, 由 PEP 向 PDP 发送的一个请求。

3.2.17 disjunctive sequence 分离的序列: 使用逻辑“或”操作组合起来的判断序列。

3.2.18 effect 结果: 满足规则的期望的结果 (或者是“Permit”或者是“Deny”)。

3.1.19 environment 环境: 与授权的决定相关但独立于特定主体、资源或动作的属性集。

3.2.20 HasPrivilegesOfRole policy HasPrivilegesOfRole 策略: 允许支持询问一个主体是否针对某一特定角色“具有特权”的包含在许可<PolicySet>中的<Policy>的选用类型。

- 3.2.21 hierarchical resource 层次资源:** 将称为节点的单独资源按照树或森林（定向无环的图）来组织的一种资源。
- 3.2.22 junior role 下级角色:** 在角色体系结构中，若角色 B 继承了与角色 A 相关联的所有许可，那么角色 A 是角色 B 的“下级”。
- 3.2.23 multi-role permissions 多角色许可:** 为得到接入，一个用户必须同时持有不仅一个角色的许可集。
- 3.2.24 named attribute 命名的属性:** 由属性名和类型、属性持有者（可以的类型：主体、资源、动作或环境）以及（选用的）签发授权的身份确定的属性特定场景。
- 3.2.25 node 节点:** 作为层次资源部分的单独资源。
- 3.2.26 obligation 职责:** 在应由 PEP 连同授权决定执行的策略或策略集规定的操作。
- 3.2.27 owner 拥有者:** 个体可确认的信息主体。
- 3.2.28 permission 许可:** 可能仅在某种特定条件情况下，对某些资源执行某些动作的能力或权利。
- 3.2.29 permission <policy set> (PPS) 许可<policy set> (PPS):** 包含有与给定角色相关联的实际许可的一个<PolicySet>。
- 3.2.30 policy information point (PIP) 策略信息点 (PIP):** 作为属性值资源动作的系统实体。
- 3.2.31 policy set 策略集:** 策略、其他策略集、组合策略算法和（选用的）职责集的集合。可以是另一个策略集的构件。
- 3.2.32 predicate 判断:** 有关其事实可被赋值的属性的声明。
- 3.2.33 resource 资源:** 数据、服务或系统构件。
- 3.2.34 role 角色:** 在具有与授权机构和分配到该角色的赋予用户责任相关语义的一个组织的上下文中的工作功能。
- 3.2.35 role based access control (RBAC) 基于角色的接入控制 (RBAC):** 用于控制接入到资源的模型，该模型中，允许针对用角色而不用单独的主体身份标识的资源进行操作。
- 3.2.36 role enablement authority 角色授权机构:** 在用户会话期间，向用户分配角色属性和值或使角色属性和值起作用的实体。
- 3.2.37 role <PolicySet> (RPS) 角色<PolicySet> (RPS):** 与具有包含与给定角色相关联的实际许可的许可<PolicySet>的给定角色属性和值的持有者相关的<PolicySet>。
- 3.2.38 senior role 高级角色:** 在角色体系结构中，若角色 A 继承了与角色 B 相关联的所有许可，那么角色 A 是角色 B 的高级角色。
- 3.2.39 rule 规则:** 目标，效果和条件。策略的构件。
- 3.2.40 rule-combining algorithm 组合规则算法:** 用于由多规则组合决定的规程。
- 3.2.41 subject 主体:** 其属性可以被判断引用的一个动作者。
- 3.2.42 target 目标:** 用于由希望赋值的资源、主体和动作定义所标识的决定请求集。
- 3.2.43 type unification 类型统一:** “统一”两种类型表达式的方法。类型表达式通过它们的结构相匹配。当类型变量出现在一个表达式中，它然后“统一”表示其他表达式的相应结构单元，无论是另一个变量或子表达式。所有变量分配必须在两个结构中保持一致。若两个表达式不能对准，或者具有不同的结构或者由于具有场景冲突，如一个变量需要表示“**xs:string**”和“**xs:integer**”，统一失败。

4 缩写

本建议书采用下列缩写：

AA	属性机构
ASP	应用服务提供者
CA	认证机构
CMP	证书管理协议
CRL	证书撤销一览表
ECP	增强的客户端/代理
HTTP	超文本传送协议
ID	标识符
IPSEC	IP安全协议
LDAP	轻便式号码簿接入协议
PAP	策略管理点
PDP	策略决定点
PEP	策略执行点
PIP	策略信息点
PKI	公钥基础设施
POP	拥有证明
PPS	许可<Policy Set>
RA	注册机构
RBAC	基于角色的接入控制
RPS	角色<PolicySet>
RSA	Rivest、Shamir和Adleman（公钥算法）
SAML	安全声明标记语言
SP	服务提供商
SSO	单一登录
TLS	传输层安全性
URI	统一资源识别符
URN	统一资源名称
XML	可扩展标记语言
XPath	XML路径语言
XSLT	可扩展的式样页语言

5 惯例

本建议书使用关键词“必须”，“必须不”、“必需”、“应该”、“不应该”、“应”、“不应”、“建议”、“可以”以及“可选”。在本建议书中，这些术语要按照 RFC 2119 中的描述来解释。

在本建议书中，短语“标准的，但可选”意味着对符合 XACML 实现，描述的功能性是可选的，但是，若声明该功能性按照本轮廓支持，那么应以所描述的方式支持它。

在句法的描述中，尖括弧("<", ">")中的单元将由合适的值来替代，方括弧("[", "]")放入可选单元，引号中的单元是文字构件，并且"*"标识前面的单元可以出现零次或多次。

6 概述

“规模化经济”已经驱使计算平台销售商开发具有非常通用功能的产品，这样这些产品应用于可能的最大范围。“在不受限制的情况下”，这些产品具有接入数据和执行软件的最大可能的特权，因而，它们可以应用于尽可能多的应用环境中，包括具有可允许的安全策略的那些应用环境。在更普遍的相对受限的安全策略的情况下，平台固有的特权必须由配置来限制。

大企业的安全策略具有许多元素和许多遵守点。策略元素可以由信息系统部、人力资源、法律部门以及经济部门来管理。策略可以由固有的执行允许的安全策略的平台 extrant、mail、WAN 以及远端接入系统来增强。为尽可能精确地实现安全策略，当前实现是管理每一个独立执行点的配置。因此，修改安全策略是昂贵和不可靠的建议。这样，在整个企业范围内有效地执行该策略来获得统一的安全措施的观点几乎是不可能的。同时，在保护企业和其客户的信息资产证明“最优方法”方面，对企业和政府执行者来讲，来自消费者、股东以及规则维护者的压力会逐渐增加。

由于这些原因，为表示安全策略，迫切需要通用语言。在遍及一个企业的实现中，通用策略语言允许该企业管理在其信息系统的所有构件中安全策略的所有元素的执行。管理安全策略可以包括下列步骤的部分或全部：写、浏览、测试、通过、签发、组合、分析、修改、丢弃、检索以及执行策略。

7 XACML核心

本部分规定了 XACML 的基础，包括通用的策略需求、模型、通用的上下文、策略句法，并且提供了一些实例。

7.1 背景

本部分为资料性的。

作为通用安全策略语言的基础，XML 是一个很自然的选择，因为它的句法和语义很容易被扩展，从而适应一个应用的特殊需求，而且它还得到了所有主流平台和工具厂商的广泛支持。

7.1.1 需求

用于表达信息系统安全策略的策略语言的基本需求如下：

- 提供把单独的规则和策略组合到应用于特定的决定请求的单一策略集中的方法。
- 提供对组合规则和策略所需流程进行灵活定义的方法。
- 提供处理具有不同动作能力的多个主体的方法。
- 提供根据主体的属性和资源做出授权决定的方法。
- 提供处理多值属性的方法。
- 提供根据信息资源的内容而做出授权决定的方法。
- 根据主体、资源和环境的属性提供逻辑和数学运算符集。
- 在提炼出定位、检索认证策略时，提供处理策略构件的分布式集的方法。
- 根据主体、资源和动作的属性值，提供快速标识应用于给定动作的策略的方法。
- 提供把策略书写者和应用环境的细节隔离开的抽象层。
- 提供具体定义一个动作集的方法，在和策略执行相组合的时候，必须执行该集动作。

XACML 的动机是采用 XML 扩展语言的接入控制策略领域中表达这些已经得到很好确定的思想。在接下来的各节中将讨论用于这些需求中的 XACML 解决方案。

7.1.2 规则和策略的组合

适用于一个特定决定请求的完整策略可以由多个单独的规则或策略组成。例如，一个个人的保密应用，个人信息的所有者可以规定信息披露的某些方面，而作为信息管理者的企业可以规定信息披露的某些其他方面。为了得到授权决定，必须有能够将两个独立的策略组合起来形成一个单一的适用于请求的策略。

XACML 规定了 3 个最高级的策略元素：`<Rule>`、`<Policy>`和`<PolicySet>`。`<Rule>`元素包含一个能够独自被赋值的布尔表达式，但是此布尔表达式不能被 PDP 独自访问。因此，它自身无法形成授权决定的基础。只有在 XACML PAP 中它可以以独自的状态存在，此时它可以构成管理的基本单元，而且可以在多个策略中被再利用。

`<Policy>`元素包含一个`<Rule>`元素集和一个组合它们赋值结果的流程。它是 PDP 所采用策略的基本单元，因此它可以形成一个授权决定的基础。

`<PolicySet>`元素包含一个`<Policy>`集或其它的`<PolicySet>`元素和一个组合这些赋值结果的特定规程。标准的方法是将独立的策略组合成单一的组合策略。

7.1.3 组合算法

XACML 规定了许多组合算法，这些组合算法可以分别通过`<Policy>`或者`<PolicySet>`元素的一个 `RuleCombiningAlgId` 或者 `PolicyCombiningAlgId` 属性来标识。规则组合算法规定了得到授权决定的流程，假定赋值一组规则的每个结果。同样的，策略组合算法规定了得到一个授权决定的流程，假定赋值一组策略的每个结果。标准的组合算法规定用于：

- Deny-overrides（有序的或者无序的）；
- Permit-overrides（有序的或者无序的）；
- First-applicable；和
- Only-one-applicable。

在拒绝主导算法下，如果遇到一个`<Rule>`或者`<Policy>`元素被赋值为"Deny"，那么无需考虑应用策略中其它`<Rule>`或者`<Policy>`的赋值结果，组合的结果为"Deny"。

用样地，在 Permit-overrides 算法中，如果遇到一个"Permit"结果，那么组合的结果就是"Permit"。

在"First-applicable"组合算法中，组合结果和赋值规则列表中第一个`<Rule>`、`<Policy>`或者`<PolicySet>`元素的结果相同，它们的目的是适用于决定请求。

"Only-one-applicable"策略组合算法只适用于策略。这个组合算法的结果是保证根据它们的目标，有一个而且只有一个策略或者策略集适用。如果没有策略或者策略集适用，那么结果为"NotApplicable"，但是如果有多于一个策略或者策略集适用，那么结果为"Indeterminate"。当只有一个策略或者策略集适用的时候，组合算法的结果是赋值一个适用策略或者策略集的结果。

策略或者策略集可以使用会改变组合算法动作的参数。但是，没有一个标准组合算法会受到参数的影响。

如果必要，本建议书的用户可以规定它们自己的组合算法。

7.1.4 多个主体

访问控制策略通常会对多个主体的动作提出要求。例如，一个管理完成较高价值财务往来的策略可以需要动作能力不同的多个个体的批准。因此，XACML 就会识别出不止一个和决定请求相关联的主体。通过一个被称作“主体类别”的属性来区分具有不同能力的主体。已经规定了一些用于这个属性的标准值，用户可以规定附加值。

7.1.5 基于主体和资源属性的策略

另外一个通用的需求就是根据主体的一些特征而不是它的身份来做出一个授权决定。或许，这个想法最常见的应用是主体角色。XACML 为支持这个方法提供了工具。可以通过<SubjectAttributeDesignator>元素标识包含在请求上下文中的主体属性。这个元素包含了一个标识属性的 URN。可选地，<AttributeSelector>元素可以在请求上下文中包含一个 XPath 表达式，通过它在上下文中的位置来标识一个特定的主体属性值。

XACML 提供了一个标准方法来引用 IETF RFC 2253 中规定的属性。这样做的目的是鼓励执行者为某些通用的主体属性采用标准的属性标识符。

另外一个通用的需求是根据资源的某些特性而不是它的身份来做出一个授权决定。XACML 提供了支持这个方法的工具。可以通过<ResourceAttributeDesignator>元素来标识资源的属性。这个元素包含一个标识属性的 URN。可选的，<AttributeSelector>元素可以在请求上下文中包含一个 XPath 表达式，通过它在上下文中的位置来标识一个特定的资源属性值。

7.1.6 多值属性

通信属性 (LDAP、XPath、SAML 等等) 的最常见的技术支持每个属性具有多个值。因此，在一个 XACML PDP 检索一个指定属性的时候，结果可以包含多个值。这些值的一个集合叫做一个包。包和集的不同在于，包可以包含重复的值，而集则不可以包含重复的值。有些时候，这种情况代表一个差错。有些时候，如果任何一个属性值满足了规则中描述的标准，那么就满足了 XACML 的规则。

XACML 提供了一个函数集，通过这个函数集，一个策略的书写者可以完全明白 PDP 如何处理多个属性值的情况。这就是“高阶”功能。

7.1.7 基于资源内容的策略

在许多应用中，需要根据请求要访问的信息资源中所包含的数据来得出一个授权决定。例如，保密策略的一个常见构件是应该允许人们阅读记录，此时对于这些记录来讲，他或者她就是主体。相应的策略必须包含一个信息资源本身所标识的主体的参考。

当可以把信息资源表示为一个 XML 文档时，XACML 提供相应的工具来支持这种做法。<AttributeSelector>元素可以在请求上下文中包含一个 XPath 表达式来标识将要在策略赋值中使用的信息资源中的数据。

7.1.8 运算符

信息安全策略根据主体的属性、资源、动作和环境来执行从而达成一个授权决定。在达成一个授权决定的过程中，可能需要比较或者计算许多不同类型的属性。例如，在一个财务应用中，需要通过把一个人的信贷限额添加到他的账户收支里边，从而对他的可用信用度进行计算。还可能要把计算结果和交易值进行比较。这种情况引发了针对主体（账户收支和信用限额）和资源（交易值）属性的数学运算的需求。

更为普遍的是，一个策略可以标识允许执行某种动作的角色集。相应的操作包含检查被主体占用的角色集和策略中标识出来的角色集之间是否有非空的交集。因此，需要集合运算。

XACML 包括许多内嵌功能和一种增加非标准功能的方法。这些功能可以嵌套用来建立任意复杂的表达式。这可以通过<Apply>元素来完成。<Apply>元素具有一个被称作 FunctionId 的 XML 属性，它用来标识应用到元素内容上的函数。每个标准函数都是为特定的参数数据类型组合而定义的，并且还规定了它的返回数据类型。因此，在书写策略或者解析策略的时候可以检查策略的数据类型的一致性。而且，可以根据策略所期望的值来检查出现在请求上下文中的数据值的类型从而保证一个可预期的结果。

除了用于数字和变量集的运算符，也为日期、时间和周期变量规定了运算符。

还为许多数据类型规定了关联运算符（相等或者对立），包括 IETF RFC 822 和 X.500 命名格式、字符串和 URI 等。

还有一个值得注意的是针对布尔数据类型的运算符，它允许一个规则中判定的逻辑组合。例如，一个规则可以包含一个声明，该声明表示在工作期间，从一个商业驻地的终端才可以允许接入。

7.1.9 策略分布

在一个分布式系统中，单独的策略声明可能会由几个策略书写者来完成并且在几个执行点上强制执行。除了帮助收集并且组合独立的策略构件，在需要的时候，这个方法还允许更新策略。XACML 策略声明可以以任何一种方式分布。然而，XACML 没有给出完成这个任务的标准方法。不考虑分布的方法，通过检查策略的<Target>元素，PDP 期望可以确认策略适用于正在处理的决定请求。

可以把<Policy>元素附加到它们应用的信息资源上。而且，可以在一个或者多个地点维持<Policy>元素，在这些地点，它们可以重新获得赋值。在这种情况下，可能要通过一个和信息资源密切相关的标识符或者定位器来引用应用的策略。

7.1.10 策略索引

为了赋值的有效性和管理的简化，可以把在一个企业内大规模采用的全部的安全策略表示为多个独立的策略部分。在这种情况下，有必要标识并且重新获得可用的策略声明，并在赋值它之前验证对于请求动作的合适性。这是 XACML 中<Target>元素的用途。

支持两种方法：

- 1) 可以把策略声明存储在一个数据库中。在这种情况下，PDP 应该形成一个数据库查询来重新获得适用于它期望应答的决定请求集的正确策略。而且，PDP 应该赋值重新获得的策略或者策略集声明的<Target>元素。
- 2) 还有一种方式就是 PDP 可以调出所有的可用策略并且在一个特定的决定请求上下文中赋值它们的<Target>元素，从而标识适用于请求的策略和策略集。

7.1.11 抽象层

PEP 可以以多种形式出现。例如，一个 PEP 可以是一个远程接入网关的一部分，一个 Web 服务器的一部分或者是一个电子邮件用户代理的一部分。希望一个企业中的所有 PEP 都工作正确，或者在将来利用一个通用的格式向一个 PDP 发布一个决定请求，这是不现实的。但是，一个特定的策略可以由多个 PEP 强制使用。强迫一个策略书写者通过几种不同的方式来书写同样的策略从而适应每个 PEP 的格式需求，效率会很低。同样地，属性可能会包含在不同的封装类型里（例如，X.509 属性证书、SAML 属性声明等）。因此，需要一个将由 XACML PDP 处理的请求和应答的标准格式。这个标准格式被称作 XACML 上下文。在 XML 模式中规定了它的句法。

自然，和 XACML 一致的 PEP 会利用 XACML 上下文发布请求和接收应答。但是，若这种情况不存在，就需要一个中间步骤来在 PEP 可以理解请求/应答格式和 PDP 可以理解的 XACML 上下文格式之间进行转换。

这种方法的好处就是策略的书写和分析可以独立于它们将要执行的特定环境。

如果初始的请求/应答格式是在 XML 模式中规定的（例如，一个与 SAML 一致的 PEP），那么初始格式和 XACML 上下文之间的转换可以通过 XSLT（可扩展样式表语言转换）格式进行定义。

同样地，如果请求访问的资源是一个 XML 文档，资源本身可以包含在请求上下文中，或者通过请求上下文引用。因此，通过在策略中使用 XPath 表达式，资源中的值可以包含在策略赋值中。

7.1.12 和强制执行相组合而执行的动作

在很多应用中，除了可以替代或者可以执行的动作以外，策略还规定了必须执行的动作。XACML 提供了工具来规定必须和<Obligations>元素中的策略赋值相组合而执行的动作。在 XACML 2.0 中没有针对这些动作的标准定义。因此，为了正确的理解，需要在一个 PAP 和 PEP 之间就将要执行的策略达成双边协议。要求和 XACML 2.0 一致的 PEP 拒绝访问，除非它们理解并且可以履行所有和应用策略相关联的<Obligations>元素。向 PEP 返回用于执行的<Obligations>元素。

7.2 XACML模型

本节为资料性的。

在接下来的各节里描述了 XACML 的数据流模型和语言模型。

7.2.1 数据流模型

图 7-1 的数据流图中示出了 XACML 域中的主要参与者。

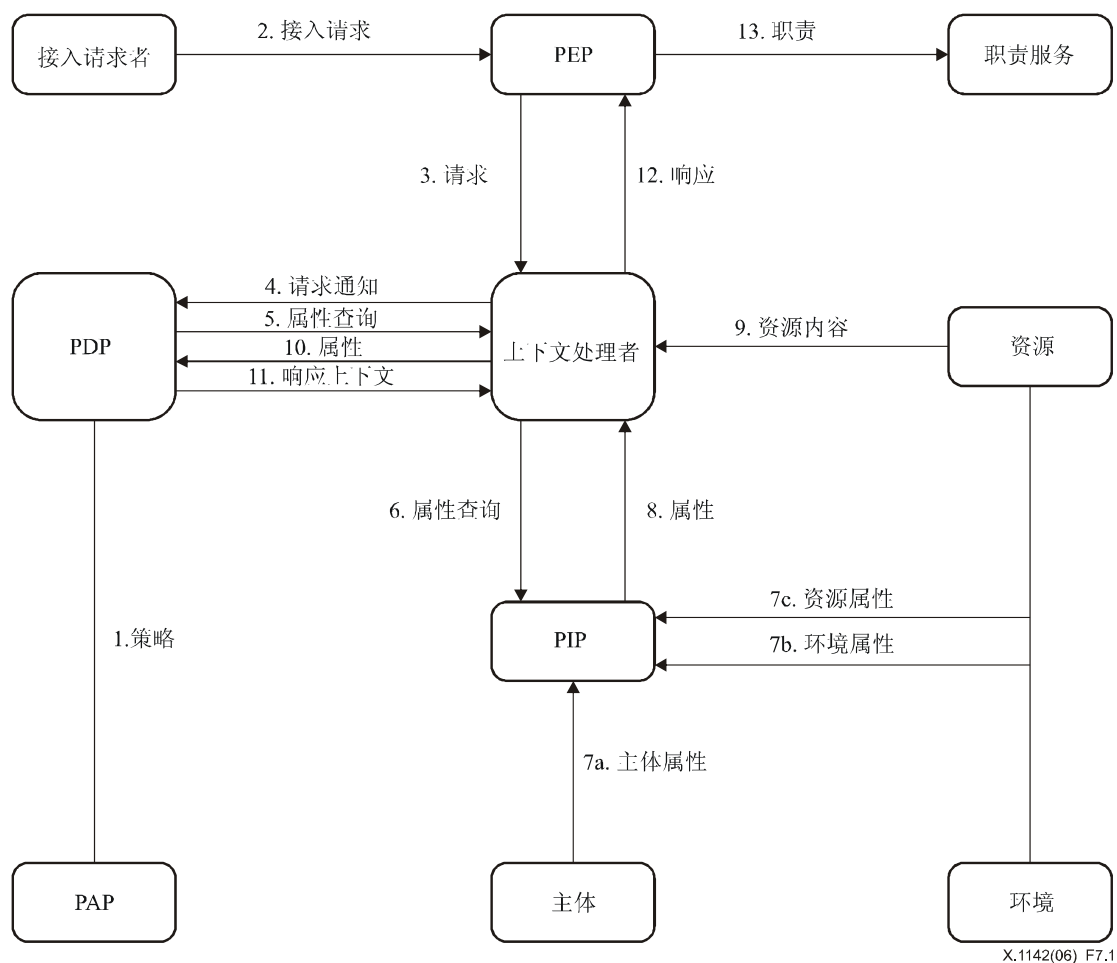


图 7-1/X.1142—数据流图示

注一 图中所示的有些数据流可以通过一个存储库来推动。例如，在上下文处理器和PIP之间的通信或者PDP和PAP之间的通信可以通过一个存储库来推动。本建议书没有有意对任何这样的存储库的位置加以限制，或者只是规定了用于任何数据流的特定的通信协议。

模型通过下面的步骤来操作：

- 1) PAP 书写策略和策略集并且使它们对PDP来讲可用。这些策略或者策略集表示用于一个规定目标的完整策略。
- 2) 接入请求者发送一个想接入PEP的请求。

- 3) PEP在它的自身请求格式中发送一个对上下文处理者的接入请求，可以选择性地包括主体、资源、动作和环境的属性。
- 4) 上下文处理者构建一个XACML请求上下文并且把它发送给PDP。
- 5) PDP可以向上下文处理者请求任何附加的主体、资源、动作和环境属性。
- 6) 上下文处理者向一个PIP请求属性。
- 7) PIP得到了请求的属性。
- 8) PIP将请求的属性返回给上下文处理者。
- 9) 上下文处理者可选择地在上下文中包含资源。
- 10) 上下文处理者把请求的属性和资源（可选）发送给PDP。PDP对策略进行赋值。
- 11) PDP把应答上下文（包括授权决定）发送给上下文处理者。
- 12) 上下文处理者把应答上下文翻译成PEP的自身应答格式。上下文处理者把应答返回给PEP。
- 13) PEP履行职责。
- 14) （没有显示出来）如果允许接入，那么PEP允许访问资源；否则，它拒绝访问资源。

7.3 XACML上下文

XACML 试图适用于不同的应用环境。通过 XACML 上下文把核心语言和应用环境隔离开来，如图 7-2 所示，阴影部分的区域指出了 XACML 的范围。在 XML 模式定义中规定了 XACML 上下文，它为 PDP 的输入和输出描述了一个标准表示法。通过一个 XACML 策略实例引用的属性可以采用上下文之上的 XPath 表达式格式，或者是通过主体、资源、动作或者环境和它们的标识符、数据类型和（可选）它的发起者来标识属性的属性指定者。执行必须在应用环境下的属性表示法（例如，SAML）和 XACML 上下文中的属性表示法之间进行转换。如何进行这种转换不在本建议书的讨论范围内。在某些情况下，例如 SAML，这种转换可以通过一个 XSLT 转换以自动方式完成（参见 W3C XSLT:1999）。

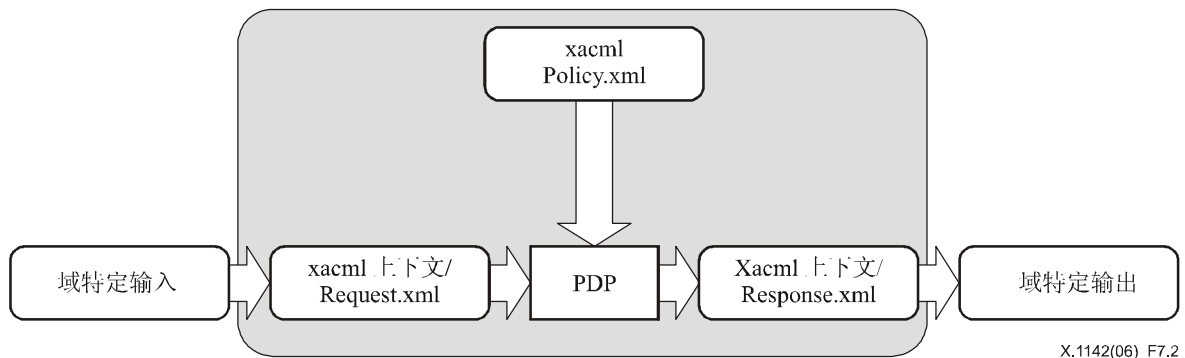


图 7-2/X.1142—XACML上下文

不要求 PDP 直接运行在一个策略的 XACML 表示法上。它可以直接操作在可替代的表示法上。

7.3.1 策略语言模型

图 7-3 示出了策略语言模型。该模型的主要构件有：

- 规则；
- 策略；和
- 策略集。

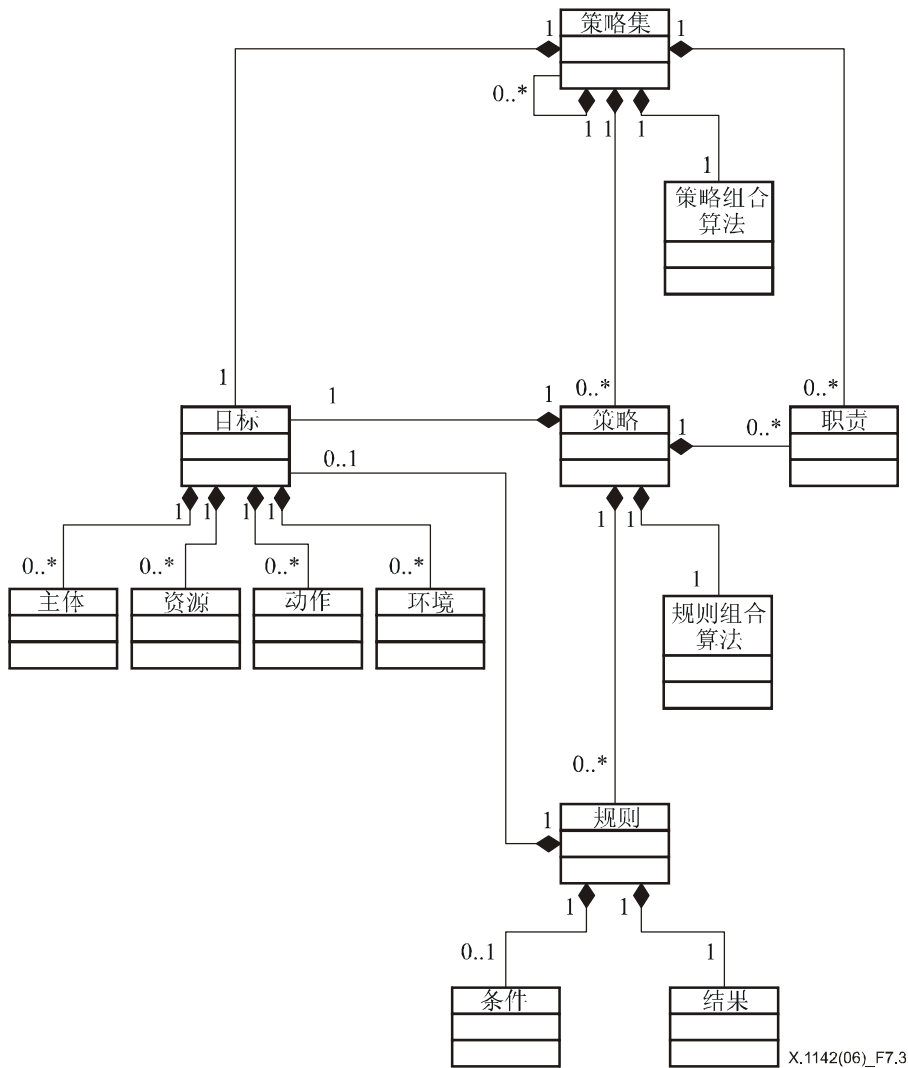


图 7-3/X.1142—策略语言模型

7.3.1.1 规则

规则是策略最基本的单元。它可以以独自的状态只存在于 XACML 域的一个主要参与者中。为了在主要参与者之间交换规则，必须把它们封装到一个策略中。可以根据一个规则的内容来对它进行赋值。一个规则的主要构件有：

- 目标；
- 结果；和
- 条件。

7.3.1.1.1 规则目标

目标规定了下面的内容集：

- 资源；
- 主体；
- 动作；和
- 环境。

规则就是要应用到上述内容。<Condition>元素可能会更进一步地提炼目标建立起来的应用。如果想要把规则应用到一个特定数据类型的所有实体，那么目标会忽略相应的实体。XACML PDP 验证请求上下文中的主体、资源、动作和环境属性可以满足由目标规定的匹配。目标的定义不连续，从而 PDP 可以有效地规定适用的规则。

一个<Rule>中可能会缺少<Target>元素。在这种情况下，<Rule>的目标和父<Policy>元素的目标相同。

特定的主体命名格式、资源命名格式和特定的资源类型是内部构建好的。例如，X.500 号码簿命名格式和 IETF RFC 822 命名格式是结构化的命名格式，然而账号没有一个可辨别的结构。UNIX 文件系统路径名和 URI 是构建资源命名格式的一个实例。而且一个 XML 文件是一个结构化资源的实例。

通常，采用结构化命名格式的一个节点（不是叶结点）的名字也是命名格式的一个合法实例。因此，例如，IETF RFC 822 名字 "med.example.com" 是一个合法的 IETF RFC 822 的名字，它用来标识一系列的通过 med.example.com 邮件服务器来定位的电子邮件地址。XPath/XPointer 值 //xacml-context:Request/xacml-context:Resource/xacml-context:ResourceContent/md:record/md:patient/ 是一个合法的 XPath/XPointer 值，它用来标识一个 XML 文件中的一个节点集。

引发的问题是：PDP 如何解释标识了一系列主体或者资源的一个名字，它是否出现在一个策略或者一个请求上下文中？它们是不是只是为了描述由名字明确标识的节点，或者它们可以表示从属于这个节点的整个子树？

在主体的情况下，没有对应这样一个节点的真正的实体。因此，这种类型的名字通常是在名字结构中从属于被标识节点的主体集。因此，非叶的主体名字不应该用在等同性函数中，只能用在匹配函数中，例如 "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match" 而不是 "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal"。

7.3.1.1.2 结果

规则的结果指示规则书写者对规则的“True”赋值的一个预期结果。存在两个值：“Permit”和“Deny”。

7.3.1.1.3 条件

条件描述了一个布尔表达式，它根据它的目标判定的声明精炼了规则的应用。因此，它可以不存在。

7.3.1.2 策略

通过数据流模型可以看出在系统实体之间没有进行规则的交换。因此，一个 PAP 把规则组合到了一个策略中。一个策略由四个主要构件组成：

- 一个目标；
- 一个组合规则的算法标识符；
- 一个规则集；和
- 职责。

7.3.1.2.1 策略目标

一个 XACML<PolicySet>、<Policy>或者<Rule>元素包含了一个<Target>元素，它规定了一系列的它适用的主体、资源、动作和环境。可以通过<PolicySet>或者<Policy>的书写者来声明一个<PolicySet>或者<Policy>的<Target>，或者可以通过它包含的<PolicySet>、<Policy>和<Rule>元素的<Target>元素来进行计算。

XACML 没有规定利用这种方式计算一个<Target>的系统实体，但是有两种逻辑方式可以采用。其中一种方式是，把外层<PolicySet>或者<Policy>（外层构件）的<Target>元素作为引用的<PolicySet>、<Policy>或者<Rule>元素（内层构件）的所有<Target>元素的一个并集来计算。另外一种方式就是，把外层构件的<Target>元素作为内层构件的<Target>元素的一个交集来计算。每种情况下的赋值结果有很大的不同：在第一种情况下，外层构件的<Target>元素适用于任何一个和至少一个内层构件的<Target>元素相匹配的决定请求；在第二种情况下，外层构件的<Target>元素只适用于和每一个内层构件的<Target>元素相匹配的决定请求。注意计算一系列<Target>元素的交集只有在数据模型相对简单的情况下具有实用性。

如果策略书写者宣布了一个<Policy>的<Target>，那么和<Policy>元素具有相同<Target>元素的<Policy>中的任何构件<Rule>元素可以忽略<Target>元素。这种<Rule>元素继承了包含它们的<Policy>中的<Target>。

7.3.1.2.2 规则组合算法

规则组合算法规定了一个流程，在赋值策略的时候，通过该流程把赋值构件规则的结果组合起来，比如，由 PDP 放置在应答上下文中的 Decision 值是策略的值，它由规则组合算法所规定。一个策略可以具有影响到规则组合算法的运算的组合参数。

7.3.1.2.3 职责

可以由策略书写者来添加职责。

当一个 PDP 赋值一个包含职责的策略，它在应答上下文中向 PEP 返回某些职责。

7.3.1.3 策略集

一个策略集由四个构件组成：

- 一个目标；
- 一个策略组合算法标识符；
- 一组策略；和
- 职责。

7.3.1.3.1 策略组合算法

策略组合算法规定了一个流程，在赋值一个策略集的时候，通过这个流程把赋值构件策略的结果组合起来，比如，由 PDP 放置在应答上下文中的 Decision 值就是赋值策略集的结果，它由策略组合算法所规定。一个策略集可以具有影响策略组合算法的运算的组合参数。

7.3.1.3.2 职责

除了包含在构件策略和策略集中的职责，一个策略集的书写者可以为策略集添加职责。

一个 PDP 赋值包含职责的策略集的时候，它在它的应答上下文中向 PEP 返回某些职责。

7.4 策略句法

下面各节中的模式定义片段是非标准的。

7.4.1 元素<PolicySet>

<PolicySet>元素是 XACML 策略模式定义中最高级的元素。<PolicySet>是其它策略集和策略的一个集合。策略集可以直接利用<PolicySet>元素或者间接采用<PolicySetIdReference>元素包含在一个封闭<PolicySet>元素中。可以直接利用<Policy>元素或者间接利用<PolicyIdReference>元素把策略放置在一个封闭的<PolicySet>元素中。

可以对<PolicySet>元素进行赋值，在这种情况下，应该采用本建议书规定的赋值流程。

如果一个<PolicySet>元素以 URL 的形式包含到其它策略集或者策略的引用，那么这些引用可以被解析。

包含在一个<PolicySet>元素中的策略集和策略必须利用由 PolicyCombiningAlgId 属性标识的算法来组合。在所有的策略组合算法中，可以把<PolicySet>看作是策略组合算法中的一个<Policy>。

<Target>元素规定了针对一系列决定请求时，<PolicySet>元素的可用性。如果<PolicySet>元素中的<Target>元素和请求上下文匹配，那么 PDP 在做出决定请求的时候可以利用<PolicySet>元素。

<Obligations>元素包含了一系列的职责，PEP 必须组合授权决定来完成这些职责。如果 PEP 无法理解或者无法完成任何一个职责，它必须按照好像 PDP 已经返回了一个“Deny”授权决定值而采取行动。

```

<xs:element name="PolicySet" type="xacml:PolicySetType"/>
<xs:complexType name="PolicySetType">
  <xs:sequence>
    <xs:element ref="xacml:Description" minOccurs="0"/>
    <xs:element ref="xacml:PolicySetDefaults" minOccurs="0"/>
    <xs:element ref="xacml:Target"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="xacml:PolicySet"/>
      <xs:element ref="xacml:Policy"/>
      <xs:element ref="xacml:PolicySetIdReference"/>
      <xs:element ref="xacml:PolicyIdReference"/>
      <xs:element ref="xacml:CombinerParameters"/>
      <xs:element ref="xacml:PolicyCombinerParameters"/>
      <xs:element ref="xacml:PolicySetCombinerParameters"/>
    </xs:choice>
    <xs:element ref="xacml:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
  <xs:attribute name="Version" type="xacml:VersionType" default="1.0"/>
  <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
</xs:complexType>

```

<PolicySet>元素是 **PolicySetType** 复杂类型。

<PolicySet>元素包含了下面的属性和元素：

- **PolicySetId** [必需]
策略集标识符。PAP 负责保证没有两个 PDP 可见的策略具有相同的标识符。可以通过下面预先规定的一个 URN 或者 URI 模式定义来完成上述需求。如果策略集标识符采用的是 URL 格式，那么它可以被解析。
- **Version** [缺省为 1.0]
PolicySet 的版本号。
- **PolicyCombiningAlgId** [必需]
策略组合算法的标识符，<PolicySet>、<CombinerParameters>、<PolicyCombinerParameters> 和 <PolicySetCombinerParameters> 构件必须通过该标识符组合起来。
- **<Description>** [可选]
策略集的一个自由格式描述。
- **<PolicySetDefaults>** [可选]
可用于策略集的缺省值集。<PolicySetDefaults>元素的范围应该是封闭的策略集。
- **<Target>** [必需]
<Target>元素针对策略请求集规定了一个策略集的可应用性。
<Target>元素可以由<PolicySet>的生成者来宣布或者可以通过引用的<Policy>元素的<Target>元素来计算，可以作为一个交集也可以作为一个并集。
- **<PolicySet>** [任意数量]
包含在这个策略集中的一个策略集。
- **<Policy>** [任意数量]
包括在这个策略集中的一个策略。
- **<PolicySetIdReference>** [任意数量]
到一个必须包含在这个策略集中的策略集的引用。如果<PolicySetIdReference>是一个 URL，那么它可以被解析。

- `<PolicyIdReference>` [任意数量]
到一个必须包含在这个策略集中的策略的一个引用。如果`<PolicyIdReference>`是一个 URL，那么它可以被解析。
- `<Obligations>` [可选]
包含`<Obligation>`元素集。
- `<CombinerParameters>` [可选]
包含`<CombinerParameter>`元素的序列。
- `<PolicyCombinerParameters>` [可选]
包含`<CombinerParameter>`元素的序列，它们和`<PolicySet>`内的一个特定的`<Policy>`或者`<PolicyIdReference>`元素相关。
- `<PolicySetCombinerParameters>` [可选]
包含`<CombinerParameter>`元素的序列，它们和`<PolicySet>`中的一个特定的`<PolicySet>`或者`<PolicySetIdReference>`元素相关。

7.4.2 元素`<Description>`

`<Description>`元素包含一个对`<PolicySet>`、`<Policy>`或者`<Rule>`元素的自由格式的描述。`<Description>`元素是 **xs:string** 简单类型。

```
<xs:element name="Description" type="xs:string"/>
```

7.4.3 元素`<PolicySetDefaults>`

`<PolicySetDefaults>`元素应该规定应用到`<PolicySet>`元素的缺省值。

```
<xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
<xs:complexType name="DefaultsType">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="xacml:XPathVersion" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

`<PolicySetDefaults>`元素是 **DefaultsType** 复杂类型。

`<PolicySetDefaults>`元素包含下列元素：

- `<XPathVersion>` [可选]
缺省的 XPath 版本。

7.4.4 元素`<XPathVersion>`

`<XPathVersion>`元素应该规定`<AttributeSelector>`元素将要采用的 W3C XPath:1999 的版本和策略集或者策略中的基于 XPath 的函数。

```
<xs:element name="XPathVersion" type="xs:anyURI"/>
```

用于 W3C XPath:1999 的 URI 是“<http://www.w3.org/TR/1999/Rec-xpath-19991116>”。如果 XACML 封闭策略集或者策略包含`<AttributeSelector>`元素或者基于 XPath 的函数，那么`<XPathVersion>`元素是必须的。

7.4.5 元素`<Target>`

`<Target>`元素标识了父元素将要赋值的决定请求集。`<Target>`元素应该以一个`<PolicySet>`和`<Policy>`元素的子元素出现或者以一个`<Rule>`元素的子元素出现。它包含了对主体、资源、动作和环境的定义。

<Target>元素包含<Subjects>、<Resources>、<Actions>和<Environments>的一个连续序列。对于可应用到决定请求的<Target>元素的父元素来讲，在<Target>元素的每个部分和<xacml-context:Request>元素的相应部分之间必须至少有一个正匹配。

```
<xs:element name="Target" type="xacml:TargetType"/>
<xs:complexType name="TargetType">
  <xs:sequence>
    <xs:element ref="xacml:Subjects" minOccurs="0"/>
    <xs:element ref="xacml:Resources" minOccurs="0"/>
    <xs:element ref="xacml:Actions" minOccurs="0"/>
    <xs:element ref="xacml:Environments" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

<Target>元素是 **TargetType** 复杂类型。

<Target>元素包含下列元素：

- <Subjects> [可选]
用于上下文中的主体属性的匹配规范，如果缺少该元素，那么目标应该匹配所有的主体。
- <Resources> [可选]
用于上下文中的资源属性的匹配规范。如果缺少该元素，那么目标应该匹配所有的资源。
- <Actions> [可选]
用于上下文中动作属性的匹配规范。如果缺少这个元素，那么目标应该匹配所有的动作。
- <Environments> [可选]
用于上下文中环境属性的匹配规范。如果缺少这个元素，那么目标应该匹配所有的环境。

7.4.6 元素<Subjects>

<Subjects>元素应该包含一个<Subject>元素的不连续序列。

```
<xs:element name="Subjects" type="xacml:SubjectsType"/>
<xs:complexType name="SubjectsType">
  <xs:sequence>
    <xs:element ref="xacml:Subject" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<Subjects>元素是 **SubjectsType** 复杂类型。

<Subjects>元素包含下列元素：

- <Subject> [一个到多个，必需]
参见7.4.7中的定义。

7.4.7 元素<Subject>

<Subject>元素应该包含一个<SubjectMatch>元素的连续序列。

```
<xs:element name="Subject" type="xacml:SubjectType"/>
<xs:complexType name="SubjectType">
  <xs:sequence>
    <xs:element ref="xacml:SubjectMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<Subject>元素是 **SubjectType** 复杂类型。

<Subject>元素包含下列元素：

- `<SubjectMatch>` [一个到多个]
请求上下文中的环境属性和嵌入属性值的单独匹配的一个连续序列。

7.4.8 元素`<SubjectMatch>`

`<SubjectMatch>`元素应该通过匹配请求上下文中`<xacml-context:Subject>`元素里的属性值与嵌入的属性值来标识一组和主体相关的实体。

```
<xs:element name="SubjectMatch" type="xacml:SubjectMatchType"/>
<xs:complexType name="SubjectMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:SubjectAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

`<SubjectMatch>`元素是 **SubjectMatchType** 复杂类型。

`<SubjectMatch>`元素包含下列属性和元素：

- `MatchId` [必需]
规定了匹配功能。这个属性的值必须是具有 7.6.5 中规定的合法值的 **xs:anyURI** 类型。
- `<xacml:AttributeValue>` [必需]
嵌入的属性值。
- `<SubjectAttributeDesignator>` [必需的选择]
可以用来标识请求上下文的`<Subject>`元素里一个或者多个属性值。
- `<AttributeSelector>` [必需的选择]
可以用来标识请求上下文中一个或者多个属性值。XPath 表达式应该可以解析为请求上下文的`<Subject>`元素里的一个属性。

7.4.9 元素`<Resources>`

`<Resources>`元素应该包含一个`<Resource>`元素的不连续序列。

```
<xs:element name="Resources" type="xacml:ResourcesType"/>
<xs:complexType name="ResourcesType">
  <xs:sequence>
    <xs:element ref="xacml:Resource" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

`<Resources>`元素是 **ResourcesType** 复杂类型。

`<Resources>`元素包含下列元素：

- `<Resource>` [一个到多个, 必需]
参见 7.4.10。

7.4.10 元素`<Resource>`

`<Resource>`元素应该包含一个`<ResourceMatch>`元素的连续序列。

```
<xs:element name="Resource" type="xacml:ResourceType"/>
<xs:complexType name="ResourceType">
  <xs:sequence>
    <xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```


<Resource>元素是 **ResourceType** 复杂类型。

<Resource>元素包含下列元素：

- <ResourceMatch> [一个到多个]
请求上下文中的资源属性和嵌入属性值的单独匹配的一个连续序列。

7.4.11 元素<ResourceMatch>

<ResourceMatch>元素应该通过匹配请求上下文中<xacml-context:Resource>元素里的属性值和嵌入的属性值来标识一组和资源相关的实体。

```
<xs:element name="ResourceMatch" type="xacml:ResourceMatchType"/>
<xs:complexType name="ResourceMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:ResourceAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

<ResourceMatch>元素是 **ResourceMatchType** 复杂类型。

<ResourceMatch>元素包含下列属性和元素：

- MatchId [必需]
规定了匹配功能。这个属性的值必须是具有 7.6.5 中定义的合法值的 **xs:anyURI** 类型。
- <xacml:AttributeValue> [必需]
嵌入的属性值。
- <ResourceAttributeDesignator> [必需的选择]
可以用来标识请求上下文<Resource>元素里的一个或者多个属性值。
- <AttributeSelector> [必需的选择]
可以用来标识请求上下文中一个或者多个属性值。XPath 表达式可以解析为请求上下文的<Resource>元素中的属性。

7.4.12 元素<Actions>

<Actions>元素应该包含一个<Action>元素的不连续序列。

```
<xs:element name="Actions" type="xacml:ActionsType"/>
<xs:complexType name="ActionsType">
  <xs:sequence>
    <xs:element ref="xacml:Action" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<Actions> 元素是 **ActionsType** 复杂类型。

<Actions>元素包含下列元素：

- <Action> [1 个到多个, 必需]
参见 7.4.13.

7.4.13 元素<Action>

<Action>元素应该包含一个<ActionMatch>元素的连续序列。

```
<xs:element name="Action" type="xacml:ActionType"/>
<xs:complexType name="ActionType">
```

```

<xs:sequence>
  <xs:element ref="xacml:ActionMatch" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

```

<Action>元素是 **ActionType** 复杂类型。

<Action>元素包含下列元素：

- <ActionMatch> [1 个到多个]

请求上下文中动作属性的独立匹配和嵌入的属性值的一个连续序列，参见 7.4.14。

7.4.14 元素<ActionMatch>

<ActionMatch>元素应该通过匹配请求上下文的<xacml-context:Action>元素中的属性值和嵌入的属性值来标识一组和动作相关联的实体。

```

<xs:element name="ActionMatch" type="xacml:ActionMatchType"/>
<xs:complexType name="ActionMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:ActionAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>

```

<ActionMatch>元素是 **ActionMatchType** 复杂类型。

<ActionMatch>元素包含下列属性和元素：

- MatchId [必需]

规定了一个匹配功能。这个属性的值必须是具有 7.6.5 中定义的合法值的 **xs:anyURI** 类型。

- <xacml:AttributeValue> [必需]

嵌入的属性值。

- <ActionAttributeDesignator> [必需的选择]

可以用来标识请求上下文的<Action>元素中的一个或者多个属性值。

- <AttributeSelector> [必需的选择]

可以用来标识请求上下文中的一个或者多个属性。XPath 表达式应该可以解析为上下文的<Action>元素中的一个属性。

7.4.15 元素<Environments>

<Environments>元素应该包含<Environment>元素的一个不连续序列。

```

<xs:element name="Environments" type="xacml:EnvironmentsType"/>
<xs:complexType name="EnvironmentsType">
  <xs:sequence>
    <xs:element ref="xacml:Environment" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

```

<Environments>元素是 **EnvironmentsType** 复杂类型。

<Environments>元素包含下列元素：

- <Environment> [1 个到多个，必需]

参见 7.4.16。

7.4.16 元素<Environment>

<Environment>元素应该包含<EnvironmentMatch>元素的一个连续序列。

```
<xs:element name="Environment" type="xacml:EnvironmentType"/>
<xs:complexType name="EnvironmentType">
  <xs:sequence>
    <xs:element ref="xacml:EnvironmentMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<Environment>元素是 **EnvironmentType** 复杂类型。

<Environment> 元素包含下列元素：

- <EnvironmentMatch> [1 个到多个]
请求上下文中的环境属性和嵌入属性值的单独匹配的一个连续序列。

7.4.17 元素<EnvironmentMatch>

<EnvironmentMatch>元素应该通过匹配请求上下文的<xacml-context:Environment>元素中的属性值和嵌入的属性值来标识一个环境。

```
<xs:element name="EnvironmentMatch" type="xacml:EnvironmentMatchType"/>
<xs:complexType name="EnvironmentMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:EnvironmentAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

<EnvironmentMatch>元素是 **EnvironmentMatchType** 复杂类型。

<EnvironmentMatch>元素包含下列属性的元素：

- MatchId [必需]
规定了一个匹配功能。这个属性值必须是具有 7.6.5 中定义的合法值的 **xs:anyURI** 类型。
- <xacml:AttributeValue> [必需]
嵌入的属性值。
- <EnvironmentAttributeDesignator> [必需的选择]
可以用来标识请求上下文的<Environment>元素中的一个或者多个属性值。
- <AttributeSelector> [必需的选择]
可以用来标识请求上下文中的一个或者多个属性值。XPath 表达式可以解析为请求上下文的<Environment>元素中的一个属性。

7.4.18 元素<PolicySetIdReference>

应该利用 <PolicySetIdReference> 元素通过 id 来引用一个 <PolicySet> 元素。如果 <PolicySetIdReference> 是一个 URL，那么它可以被解析为 <PolicySet> 元素。但是，把一个策略集引用解析为相应的策略集的机制不在本建议书的讨论范围内。

```
<xs:element name="PolicySetIdReference" type="xacml:IdReferenceType"/>
<xs:complexType name="IdReferenceType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="xacml:Version" type="xacml:VersionMatchType"
        use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

```

        <xs:attribute name="xacml:EarliestVersion"
type="xacml:VersionMatchType" use="optional"/>
        <xs:attribute name="xacml:LatestVersion"
type="xacml:VersionMatchType" use="optional"/>
    </xs:extension>
</xs:simpleContent>
</xs:complexType>

```

<PolicySetIdReference>元素是 **xacml:IdReferenceType** 复杂类型。

IdReferenceType 通过下面的属性对 **xs:anyURI** 类型进行了扩展：

- **Version** [可选]
为引用的策略集版本规定了匹配表达式。
- **EarliestVersion** [可选]
为引用的策略集最新可接受版本规定了一个匹配表达式。
- **LatestVersion** [可选]
规定了用于引用策略集的最新可接受版本的一个匹配表达式。

在<PolicySetIdReference>中可能会出现这些属性的任何一种组合。引用的策略集必须匹配所有的表达式。如果这些属性都没有出现，那么策略集的任何版本都可以接受。如果可以得到多个匹配版本，那么应该采用最新的版本。

7.4.19 元素<PolicyIdReference>

应该利用 <xacml:PolicyIdReference> 元素通过 **id** 来引用一个 <Policy> 元素。如果 <PolicyIdReference> 是一个 URL，那么它可以被解析为 <Policy> 元素。但是，解析一个策略引用到相应的策略的机制不在本建议书的讨论范围内。

```
<xs:element name="PolicyIdReference" type="xacml:IdReferenceType"/>
```

<PolicyIdReference>元素是 **xacml:IdReferenceType** 复杂类型。

7.4.20 简单类型VersionType

这个类型的元素应该包含策略或者策略集的版本号。

```

<xs:simpleType name="VersionType">
  <xs:restriction base="xs:string">
    <xs:pattern value="(\d+\.)*\d+"/>
  </xs:restriction>
</xs:simpleType>

```

将版本号表示为一个十进制数序列，每个数字之间通过一个点（.）隔开。'd+'代表一个或者多个小数位数的一个序列。

7.4.21 简单类型VersionMatchType

这个类型的元素应该包含一个和版本号相匹配受限的常规表达式。表达式应该匹配一个被引用的策略或者策略集的版本，这些策略或者策略集可以包含到正在引用的策略或者策略集中。

```

<xs:simpleType name="VersionMatchType">
  <xs:restriction base="xs:string">
    <xs:pattern value="((\d+|\*)\.)*(\d+|\*|\+)" />
  </xs:restriction>
</xs:simpleType>

```

版本匹配和版本串一样都通过“.”来隔开。数字代表一个直接的数字匹配。'*'表示任何一个数字都有效。一个 '+' 意味着任何数字，和任何后续的数字都有效。在这种方式下，下面的四个图案应该都匹配版本串 '1.2.3': '1.2.3', '1.*.3', '1.2.*' 和 '1.+'

7.4.22 元素<Policy>

<Policy>元素是应该提交给 PDP 进行赋值的最小的实体。

可以对一个 <Policy> 进行赋值，在这种情况下，应该采用 7.6.10 中定义的赋值流程。

这个元素的主要构件是<Target>、<Rule>、<CombinerParameters>、<RuleCombinerParameters>和<Obligations>元素和 RuleCombiningAlgId 属性。

<Target>元素规定了<Policy>元素对决定请求集的可应用性。如果<Policy>元素中的<Target>元素和请求上下文相匹配，那么 PDP 在做授权决定的时候，可以利用<Policy>元素。

<Policy>元素包含了一个在<VariableDefinition>和<Rule>元素之间选择的序列。

必须通过由 RuleCombiningAlgId 属性定义的算法把<Policy>元素中包含的规则组合起来。

<Obligations>元素包含一组职责，PEP 必须联合授权决定完成这些职责。

```
<xs:element name="Policy" type="xacml:PolicyType"/>
<xs:complexType name="PolicyType">
  <xs:sequence>
    <xs:element ref="xacml:Description" minOccurs="0"/>
    <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
    <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>
    <xs:element ref="xacml:Target"/>
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>
      <xs:element ref="xacml:RuleCombinerParameters" minOccurs="0"/>
      <xs:element ref="xacml:VariableDefinition"/>
      <xs:element ref="xacml:Rule"/>
    </xs:choice>
    <xs:element ref="xacml:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
  <xs:attribute name="Version" type="xacml:VersionType" default="1.0"/>
  <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

<Policy>元素是 **PolicyType** 复杂类型。

<Policy>元素包含下列属性和元素：

- PolicyId [必需]
策略标识符。PAP 负责保证没有两个 PDP 可见的策略具有相同的标识符。可以通过一个预先规定的 URN 或者 URI 模式来完成上面的需求。如果策略标识符采用的是 URL 格式，那么它可以被解析。
- Version [缺省为 1.0]
策略的版本号。
- RuleCombiningAlgId [必需]
规则组合算法标识符，<Policy>、<CombinerParameters>和 <RuleCombinerParameters>构件必须通过它来进行组合。
- <Description> [可选]
策略的一个自由格式的描述。
- <PolicyDefaults> [可选]
定义了适应于策略的一组缺省值。<PolicyDefaults>元素的范围应该是封闭的策略。
- <CombinerParameters> [可选]
规则组合算法采用的一个参数序列。
- <RuleCombinerParameters> [可选]
规则组合算法采用的一个参数序列。

- <Target> [必需]
<Target>元素定义了<Policy>对决定请求集的可应用性。
可以通过<Policy>元素的生成者来发布<Target>元素，或者可以作为一个交集或者一个并集，通过被引用的<Rule>元素的<Target>元素计算出来。
- <VariableDefinition> [任意数量]
从可以发现表达式的一个规则内的任何地点引用的公共的函数定义。
- <Rule> [任意数量]
根据 RuleCombiningAlgId 属性必需组合起来的一个规则序列。必需考虑<Target>元素和决定请求相匹配的规则。应该忽略<Target>元素和决定请求不匹配的规则。
- <Obligations> [可选]
必需由 PEP 组合授权决定来完成的职责的一个连续序列。

7.4.23 元素<PolicyDefaults>

<PolicyDefaults>元素应该规定应用到<Policy>元素的缺省值。

```
<xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
<xs:complexType name="DefaultsType">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="xacml:XPathVersion" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

<PolicyDefaults>元素是 **DefaultsType** 复杂类型。

<PolicyDefaults>元素包含下列元素：

- <XPathVersion> [可选]
缺省 XPath 版本。

7.4.24 元素<CombinerParameters>

<CombinerParameters>元素为一个策略或者规则组合算法传送参数。

如果多个<CombinerParameters>元素出现在同样的策略或者策略集中，应该认为它们等同于包含了所有包含在上述<CombinerParameters>元素中的<CombinerParameters>所有序列的串联的一个<CombinerParameters>元素，因此在<CombinerParameter>元素的串联中保持着<CombinerParameters>元素出现的顺序。

注意在 XACML2.0 中定义的组合算法都不是参数化的。

```
<xs:element name="CombinerParameters" type="xacml:CombinerParametersType"/>
<xs:complexType name="CombinerParametersType">
  <xs:sequence>
    <xs:element ref="xacml:CombinerParameter" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<CombinerParameters>元素是 **CombinerParametersType** 复杂类型。

<CombinerParameters>元素包含下列元素：

- <CombinerParameter> [任意数量]
一个单一参数。

对<CombinerParameters>元素的支持为可选。

7.4.25 元素<CombinerParameter>

<CombinerParameter>元素为一个策略或者规则组合算法传递一个单一参数。

```
<xs:element name="CombinerParameter" type="xacml:CombinerParameterType"/>
<xs:complexType name="CombinerParameterType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
  </xs:sequence>
  <xs:attribute name="ParameterName" type="xs:string" use="required"/>
</xs:complexType>
```

<CombinerParameter>元素是 **CombinerParameterType** 复杂类型。

<CombinerParameter>元素包含下列属性：

- ParameterName [必需]
参数的标识符。
- AttributeValue [必需]
参数值。

对<CombinerParameter>元素的支持为可选。

7.4.26 元素<RuleCombinerParameters>

<RuleCombinerParameters>元素为一个规则组合算法传递一个策略中和一个特定的规则相关联的参数。

每个<RuleCombinerParameters>元素必须和一个包含在同一个策略中的规则相关联。如果多个<RuleCombinerParameters>元素引用同样的规则，那么应该认为它们等同于包含了包含在上述<RuleCombinerParameters>元素中的<CombinerParameters>所有序列的串联的一个<RuleCombinerParameters>元素，因此在<CombinerParameter>元素的串联中保持了<RuleCominberParameters>元素出现的顺序。

注意在 XACML2.0 中定义的规则组合算法没有参数化的。

```
<xs:element name="RuleCombinerParameters"
type="xacml:RuleCombinerParametersType"/>
<xs:complexType name="RuleCombinerParametersType">
  <xs:complexContent>
    <xs:extension base="xacml:CombinerParametersType">
      <xs:attribute name="RuleIdRef" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

<RuleCombinerParameters>元素包含下列元素：

- RuleIdRef [必需]
包含在策略中的<Rule>的标识符。

只有在没有实现对组合参数的支持的情况下，对<RuleCombinerParameters>元素的支持为可选。

7.4.27 元素<PolicyCombinerParameters>

<PolicyCombinerParameters>元素为一个策略组合算法传递和一个策略集中的特定策略相关联的参数。

每个<PolicyCombinerParameters>元素必须和包含在同样策略集中的一个策略相关联。如果多个<PolicyCombinerParameters>引用同样的策略，那么应该认为它们等同于包含了包含在所有上述<PolicyCombinerParameters>元素中的<CombinerParameters>所有序列的串联的一个<PolicyCombinerParameters>元素，这样在<CombinerParameter>元素的串联中保持了<PolicyCombinerParameters>元素的出现顺序。

注意在 XACML 2.0 中定义的策略组合算法没有参数化的。

```
<xs:element name="PolicyCombinerParameters"
type="xacml:PolicyCombinerParametersType"/>
<xs:complexType name="PolicyCombinerParametersType">
  <xs:complexContent>
    <xs:extension base="xacml:CombinerParametersType">
      <xs:attribute name="PolicyIdRef" type="xs:anyURI"
use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

<PolicyCombinerParameters>元素是 **PolicyCombinerParametersType** 复杂类型。

<PolicyCombinerParameters>元素包含下列元素：

— PolicyIdRef [必需]

一个<Policy>的标识符或者策略集中的一个<PolicyIdReference>的值。

只有在没有执行对组合参数的支持的情况下，对<PolicyCombinerParameters>元素的支持为可选。

7.4.28 元素<PolicySetCombinerParameters>

<PolicySetCombinerParameters>元素为一个策略组合算法传递和一个策略集中一个特定策略相关的参数。

每个<PolicySetCombinerParameters>元素必须和一个包含在同样一个策略集内的一个策略相关联。如果多个<PolicySetCombinerParameters>元素引用同样的策略集，那么应该认为它们等同于包含了包含在上述 <PolicySetCombinerParameters> 元素中的 <CombinerParameters> 所有序列的串联的一个 <PolicySetCombinerParameters> 元素，因此在 <CombinerParameter> 元素的串联中保持了 <PolicySetCombinerParameters>元素的出现顺序。

注意，XACML 2.0 中定义的策略算法没有参数化的。

```
<xs:element name="PolicySetCombinerParameters"
type="xacml:PolicySetCombinerParametersType"/>
<xs:complexType name="PolicySetCombinerParametersType">
  <xs:complexContent>
    <xs:extension base="xacml:CombinerParametersType">
      <xs:attribute name="PolicySetIdRef" type="xs:anyURI"
use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

<PolicySetCombinerParameters>元素是 **PolicySetCombinerParametersType** 复杂类型。

<PolicySetCombinerParameters>元素包含下列元素：

— PolicySetIdRef [必需]

一个<PolicySet>的标识符或者策略集中的一个<PolicySetIdReference>值。

只有在没有执行对组合参数的支持的情况下，对<PolicySetCombinerParameters>元素的支持是可选的。

7.4.29 元素<Rule>

<Rule>元素应该在策略中定义独立的规则。这个元素的主要构件是<Target>和<Condition>元素和 Effect 属性。

应该对<Rule>元素进行赋值，在这种情况下，应该采用 7.6.9 中定义的赋值流程。

```
<xs:element name="Rule" type="xacml:RuleType"/>
<xs:complexType name="RuleType">
  <xs:sequence>
    <xs:element ref="xacml:Description" minOccurs="0"/>
```



```

    <xs:element ref="xacml:Target" minOccurs="0"/>
    <xs:element ref="xacml:Condition" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="RuleId" type="xs:string" use="required"/>
  <xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
</xs:complexType>

```

<Rule>元素是 **RuleType** 复杂类型。

<Rule>元素包含下述属性和元素：

- RuleId [必需]
定义这个规则的一个字符串。
- Effect [必需]
规则的结果。这个属性的值是"Permit"或者"Deny"。
- <Description> [可选]
规则的一个自由格式的描述。
- <Target> [可选]
标识<Rule>元素打算赋值的决定请求集。如果忽略了这个元素，那么应该通过封闭<Policy>元素的<Target>元素来定义<Rule>的目标。详见 7.6.6。
- <Condition> [可选]
必须满足将要为规则分配 Effect 值的一个判定。

7.4.30 简单类型 EffectType

简单类型 **EffectType** 定义了可以用于<Rule>元素 Effect 属性的值，和用于<Obligation> 元素的 FulfillOn 属性的值。

```

<xs:simpleType name="EffectType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Permit"/>
    <xs:enumeration value="Deny"/>
  </xs:restriction>
</xs:simpleType>

```

7.4.31 元素<VariableDefinition>

<VariableDefinition>元素应该用来定义可以被一个<VariableReference>元素引用的值。为它的 VariableId 属性提供的名字不应该出现在所包含策略中的任何其它<VariableDefinition>元素的 VariableId 属性中。<VariableDefinition>元素可能包含了未定义的<VariableReference>元素，如果它真的包含了未定义的<VariableReference>元素，那么稍后在所包含策略中必须对相应的<VariableDefinition>元素进行定义。<VariableDefinition>元素必须集中起来或者可以放置在 encompassing 策略中离引用较近的位置。可以具有 0 个或者多个对每个<VariableDefinition>元素的引用。

```

<xs:element name="VariableDefinition" type="xacml:VariableDefinitionType"/>
<xs:complexType name="VariableDefinitionType">
  <xs:sequence>
    <xs:element ref="xacml:Expression"/>
  </xs:sequence>
  <xs:attribute name="VariableId" type="xs:string" use="required"/>
</xs:complexType>

```

<VariableDefinition>元素是 **VariableDefinitionType** 复杂类型。<VariableDefinition>元素就有下述元素和属性：

- <Expression> [必需]
ExpressionType 复杂类型的任何一个元素。

- VariableId [必需]
变量定义的名字。

7.4.32 元素<VariableReference>

<VariableReference> 元素用来引用在同样的封闭元素 <Policy> 中进行定义的一个值。<VariableReference> 元素应该通过针对它们各自 VariableId 属性值串等同性来引用 <VariableDefinition> 元素。在同样的封闭元素 <Policy> 中应该存在而且只能存在一个 <VariableReference> 可以引用的 <VariableDefinition>。可以有 0 个或者多个 <VariableReference> 元素引用同样的 <VariableDefinition> 元素。

```
<xs:element name="VariableReference" type="xacml:VariableReferenceType"
substitutionGroup="xacml:Expression"/>
<xs:complexType name="VariableReferenceType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="VariableId" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

<VariableReference> 元素是 **VariableReferenceType** 的复杂类型，**VariableReferenceType** 的复杂类型是 **ExpressionType** 的复杂类型，而且它是 <Expression> 元素替代组的一个成员。在模式定义中，<VariableReference> 元素可以出现在任何 <Expression> 元素出现的地方。

<VariableReference> 元素具有下面的属性：

- VariableId [必需]
用来引用在一个 <VariableDefinition> 元素中定义的值的名字。

7.4.33 元素<Expression>

<Expression> 元素没有直接用于一个策略中。<Expression> 元素表示扩展了 **ExpressionType** 的一个元素，同时它是将要出现在它的位置上的 <Expression> 元素替代组的一个成员。

```
<xs:element name="Expression" type="xacml:ExpressionType" abstract="true"/>
<xs:complexType name="ExpressionType" abstract="true"/>
```

下面的元素位于 <Expression> 元素替代组中：

<Apply>、<AttributeSelector>、<AttributeValue>、<Function>、<VariableReference>、<ActionAttributeDesignator>、<ResourceAttributeDesignator>、<SubjectAttributeDesignator> 和 <EnvironmentAttributeDesignator>。

7.4.34 元素<Condition>

<Condition> 元素是一个用于主体、资源、动作和环境属性的布尔函数或者属性函数。

```
<xs:element name="Condition" type="xacml:ConditionType"/>
<xs:complexType name="ConditionType">
  <xs:sequence>
    <xs:element ref="xacml:Expression"/>
  </xs:sequence>
</xs:complexType>
```

<Condition> 包含一个 <Expression> 元素，它的一个限制条件就是 <Expression> 返回的数据类型必须是 "http://www.w3.org/2001/XMLSchema#boolean"。

7.4.35 元素<Apply>

<Apply> 元素表示一个函数对它的变量的应用，因此编码一个函数调用。<Apply> 可以应用到任何 <Expression> 元素替代组成员的组合当中。

```
<xs:element name="Apply" type="xacml:ApplyType"
substitutionGroup="xacml:Expression"/>
<xs:complexType name="ApplyType">
```

```

<xs:complexContent>
  <xs:extension base="xacml:ExpressionType">
    <xs:sequence>
      <xs:element ref="xacml:Expression" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>

```

<Apply>元素是 **ApplyType** 复杂类型。

<Apply>元素包含下列属性和元素：

- **FunctionId** [必需]
可应用到变量的函数的标识符。参见附件 A 中描述的 XACML 定义的函数。
- **<Expression>** [可选]
函数的变量，这些变量可以包含其它函数。

7.4.36 元素<Function>

<Function>元素应该用来命名一个函数，正如一个参数对一个通过父元素<Apply>定义的函数的作用。如果父<Apply>函数是一个高阶包函数，那么把所命名的函数应用到通过父元素的其它参数标识出来的包的每个元素上。

```

<xs:element name="Function" type="xacml:FunctionType"
substitutionGroup="xacml:Expression"/>
<xs:complexType name="FunctionType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

Function 元素是 **FunctionType** 复杂类型。

Function 元素包含下列属性：

- **FunctionId** [必需]
函数的标识符。

7.4.37 复杂类型AttributeDesignatorType

复杂类型 **AttributeDesignatorType** 是用于通过名字来标识属性的元素类型。它包含匹配请求上下文中属性所必需的信息。

它还包括在上下文中没有出现匹配属性的情况下，用来控制动作的信息。

这个类型的元素不应该改变所命名属性的匹配语义，但是可以缩小搜索范围。

```

<xs:complexType name="AttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="AttributeId" type="xs:anyURI"
use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:attribute name="Issuer" type="xs:string" use="optional"/>
      <xs:attribute name="MustBePresent" type="xs:boolean"
use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

如果它们相应的 AttributeId、DataType 和 Issuer 属性值相匹配，那么命名的属性应该匹配一个属性。通过 URI 等同性，属性指定者 AttributeId 必须和属性的 AttributeId 相匹配。通过 URI 等同性，属性指定者的 DataType 必须和同一个属性的 DataType 相匹配。

如果在属性指定者中出现了 Issuer 属性，那么它必须通过 "urn:oasis:names:tc:xacml:1.0:function:string-equal" 函数，和同一个属性的 Issuer 相匹配。如果在属性指定者中没有出现 Issuer，那么属性和指定属性的匹配应该由 AttributeId 和 DataType 属性单独来管理。

<AttributeDesignatorType> 包含下列属性：

- AttributeId [必需]
该属性应该规定用来匹配属性的 AttributeId。
- DataType [必需]
由 <AttributeDesignator> 元素返回的包应该包含这个数据类型的值。
- Issuer [可选]
如果提供了该属性，那么它应该规定用来匹配属性的 Issuer。
- MustBePresent [可选]
在请求上下文中没有出现指定的属性的情况下，本属性用来控制元素是否返回 "Indeterminate" 或者一个空的包。

7.4.38 元素 <SubjectAttributeDesignator>

<SubjectAttributeDesignator> 元素从一个请求上下文中为一个指定的分类主体重新获取一个值的包。一个主体属性是一个包含在请求上下文的 <Subject> 元素中的一个属性。一个分类主体是由一个特定主体类别属性来标识的主体。一个指定的分类主体属性是用于一个特定的分类主体的一个指定的主体属性。

<SubjectAttributeDesignator> 元素应该返回一个包，该包包含所有的通过给定的分类主体属性相匹配的主体属性值。如果在上下文中没有出现匹配的属性，那么 MustBePresent 属性负责控制这个元素是否返回了一个空包或者 "Indeterminate"。

SubjectAttributeDesignatorType 扩展 **AttributeDesignatorType** 的匹配语义，因此它把属性搜索空间缩小到了特定类别主体，因此通过 URI 等同性，这个元素的 SubjectCategory 属性的值和请求上下文的 <Subject> 元素的 SubjectCategory 属性的值相匹配。

如果请求上下文包含多个具有同样 SubjectCategory XML 属性的主体，那么应该把它们看作是同一个分类主体。

<SubjectAttributeDesignator> 可以会出现在 <SubjectMatch> 元素中，而且可以作为一个变量传递给 <Apply> 元素。

```
<xs:element name="SubjectAttributeDesignator"
type="xacml:SubjectAttributeDesignatorType"
substitutionGroup="xacml:Expression"/>
<xs:complexType name="SubjectAttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:AttributeDesignatorType">
      <xs:attribute name="SubjectCategory" type="xs:anyURI"
use="optional" default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The <SubjectAttributeDesignator> 元素是 **SubjectAttributeDesignatorType** 类型。**SubjectAttributeDesignatorType** 复杂类型通过一个 SubjectCategory 属性对 **AttributeDesignatorType** 复杂类型进行了扩展。

— SubjectCategory [可选]

本属性应该规定分类主体，通过它来和指定的主体属性相匹配。如果没有出现 SubjectCategory，那么应该采用它的缺省值"urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"。

7.4.39 元素<ResourceAttributeDesignator>

<ResourceAttributeDesignator>元素从请求上下文为一个指定的资源属性重新获得一个值的包。一个资源属性是一个包含在请求上下文的<Resource>元素里的属性。一个指定的资源属性是匹配资源属性的指定的属性。如果至少有一个资源属性和下面的标准相匹配，那么应该认为出现了一个指定的资源属性。一个资源属性值是包含在一个资源属性内的属性值。

<ResourceAttributeDesignator>元素应该返回一个包含所有资源属性值的包，该包包含所有的通过指定的资源属性相匹配的资源属性值。如果在上下文中没有出现匹配的属性，MustBePresent 属性负责控制这个元素是否返回了一个空包或者"Indeterminate"。

一个指定的资源属性应该按照 **AttributeDesignatorType** 复杂类型中定义的每个匹配语义来匹配一个资源属性。

<ResourceAttributeDesignator>可以出现在<ResourceMatch>元素中并且可以作为一个变量传递给<Apply>元素。

```
<xs:element name="ResourceAttributeDesignator"
type="xacml:AttributeDesignatorType" substitutionGroup="xacml:Expression"/>
```

<ResourceAttributeDesignator>元素是 **AttributeDesignatorType** 复杂类型。

7.4.40 元素<ActionAttributeDesignator>

<ActionAttributeDesignator>元素从请求上下文为一个指定动作属性重新获得一个值的包。一个动作属性是一个包含在请求上下文的<Action>元素中的属性。一个指定的动作属性具有特定的标准（如下），通过这些标准来和一个动作属性相匹配。如果至少有一个动作属性与标准匹配，那么应该认为出现了一个指定的动作属性。一个动作属性值是包含在一个动作属性中的一个属性值。

<ActionAttributeDesignator>元素应该返回一个通过指定动作属性而匹配的所有动作属性值的包。如果中没有出现匹配属性，MustBePresent 属性负责控制这个元素是否返回了一个空包或者"Indeterminate"。

一个指定的动作属性应该按照 **AttributeDesignatorType** 复杂类型中定义的每个匹配语义来匹配一个动作属性。

<ActionAttributeDesignator>可以出现在<ActionMatch>元素中而且可以作为一个变量传递给<Apply>元素。

```
<xs:element name="ActionAttributeDesignator" type="xacml:AttributeDesignatorType"
substitutionGroup="xacml:Expression"/>
```

<ActionAttributeDesignator>元素是 **AttributeDesignatorType** 复杂类型。

7.4.41 元素<EnvironmentAttributeDesignator>

<EnvironmentAttributeDesignator>元素从请求上下文为一个指定的环境重新获取了一个值的包。一个环境属性是一个包含在请求上下文的<Environment>元素里边的一个属性。一个指定的环境属性具有特定的准则（后面描述），通过这些标准来匹配一个环境属性。如果至少有一个环境属性与准则匹配，那么应该认为存在一个指定的环境属性。一个环境属性值是包含在一个环境属性内的一个属性值。

<EnvironmentAttributeDesignator>元素应该赋值包中包含的通过指定的环境属性来匹配的所有的环境属性值。如果在上下文中没有出现匹配属性，那么 MustBePresent 属性负责控制这个元素是否返回了一个空的包或者"Indeterminate"。

一个指定的环境属性应该按照 **AttributeDesignatorType** 复杂类型中定义的每个匹配语义来匹配一个环境属性。

可以把<EnvironmentAttributeDesignator>作为一个变量传递给<Apply>元素。

```
<xs:element name="EnvironmentAttributeDesignator" type="xacml:AttributeDesignatorType"
substitutionGroup="xacml:Expression"/>
```

<EnvironmentAttributeDesignator>元素是 **AttributeDesignatorType** 复杂类型。

7.4.42 元素<AttributeSelector>

<AttributeSelector>元素通过属性在请求上下文的位置而标识属性。对<AttributeSelector>的支持为可选。

<AttributeSelector>元素的 RequestContextPath XML 属性应该包括一个合法的 XPath 表达式，它的上下文节点是<xacml-context:Request>元素。AttributeSelector 元素应该赋值一系列的值，由元素的数据类型属性定义了这些值的数据类型。如果在 AttributeSelector 中定义的数据类型是原语数据类型（参见 W3C Schema:2001、W3C Datatypes:2001, 3.2 中的定义），那么应该把由 XPath 表达式返回的 lexical value 转换成为<AttributeSelector>定义的数据类型值。如果一个差错导致把 XPath 表达式返回的值进行了转换，例如在值不是一个有效的数据类型实例的时候，那么<AttributeSelector>的值应该是"Indeterminate"。

```
xs:string()
xs:boolean()
xs:integer()
xs:double()
xs:dateTime()
xs:date()
xs:time()
xs:hexBinary()
xs:base64Binary()
xs:anyURI()
```

如果 AttributeSelector 中定义的数据类型不是上述原 DataTypes 中的一个，那么 AttributeSelector 应该返回一个已定义数据类型的一系列实例。如果在把由 XPath 表达式返回的值转换为已定义的数据类型时，出现了差错，那么 AttributeSelector 的结果应该是"Indeterminate"。

由已定义的 XPath 表达式选定的每个节点必须是文本节点、属性节点、处理指令节点或者注释节点中的任意一个。必须把每个节点的值串表示法转换为一个已定义数据类型的属性值，AttributeSelector 的结果是由所有选定节点生成的一系列属性值的包。

如果由已定义的 XPath 表达式选定的节点不是上述节点中的任何一个（也就是说，一个文本节点、一个属性节点、一个处理指定节点或者一个注释节点），那么封闭策略的结果应该是"Indeterminate"，它的 StatusCode 值为 "urn:oasis:names:tc:xacml:1.0:status:syntax-error"。

```
<xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"
substitutionGroup="xacml:Expression"/>
<xs:complexType name="AttributeSelectorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="RequestContextPath" type="xs:string"
use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:attribute name="MustBePresent" type="xs:boolean"
use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

<AttributeSelector>元素 **AttributeSelectorType** 复杂类型。

<AttributeSelector>元素具有下述属性：

- RequestContextPath [必需]
一个 XPath 表达式，它的上下文节点是<xacml-context:Request>元素。对 XPath 句法没有限制。

- `DataType` [必需]
由<AttributeSelector>元素返回的一系列应该包含这个数据类型的值。
- `MustBePresent` [可选]
这个属性负责控制在 XPath 表达式没有选择节点的时候，元素是否返回"Indeterminate"或者一个空。

7.4.43 元素<AttributeValue>

<xacml:AttributeValue>元素应该包含一个文字属性值。

```
<xs:element name="AttributeValue" type="xacml:AttributeValueType"
substitutionGroup="xacml:Expression"/>
<xs:complexType name="AttributeValueType" mixed="true">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:sequence>
        <xs:any namespace="##any" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

<xacml:AttributeValue>元素是 **AttributeValueType** 复杂类型。

<xacml:AttributeValue>元素具有下述属性：

- `DataType` [必需]
属性值的数据类型。

7.4.44 元素<Obligations>

<Obligations>元素应该包含一组<Obligation>元素。

对<Obligations>元素的支持为可选。

```
<xs:element name="Obligations" type="xacml:ObligationsType"/>
<xs:complexType name="ObligationsType">
  <xs:sequence>
    <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<Obligations>元素是 **ObligationsType** 复杂类型。

<Obligations>元素包含下述元素：

- <Obligation> [1 个到多个]
职责的一个序列。

7.4.45 元素<Obligation>

<Obligation>元素应该包含用于职责的一个标识符和一组属性，这些属性构成了由职责定义的动作的变量。FulfillOn 属性应该指示 PEP 必须完成的职责的效果。

```
<xs:element name="Obligation" type="xacml:ObligationType"/>
<xs:complexType name="ObligationType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
  <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
</xs:complexType>
```

<Obligation> 元素是 **ObligationType** 复杂类型。若要了解如何确定 PDP 返回的职责集的描述见 7.6.14。

<Obligation>元素包含下述元素和属性:

- ObligationId [必需]
职责标识符。应该由 PEP 解释职责标识符的值。
- FulfillOn [必需]
PEP 必须完成的职责的效果。
- <AttributeAssignment> [可选]
职责变量分配。应该由 PEP 来解释职责变量的值。

7.4.46 元素<AttributeAssignment>

<AttributeAssignment>元素用于在职责中包含变量。它应该通过扩展 **AttributeValueType** 类型定义来包含一个 AttributeId 和相应的属性值。可以利用任何和模式定义句法一致的方式来使用 <AttributeAssignment>元素,它是<xs:any>元素的一个序列。PEP 应该明白定义的值,但是 XACML 没有更进一步定义该值。

```
<xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
<xs:complexType name="AttributeAssignmentType" mixed="true">
  <xs:complexContent>
    <xs:extension base="xacml:AttributeValueType">
      <xs:attribute name="AttributeId" type="xs:anyURI"
use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

<AttributeAssignment>元素是 **AttributeAssignmentType** 复杂类型。

<AttributeAssignment>元素包含下述属性:

- AttributeId [必需]
属性标识符。

7.5 上下文句法

接下来的各节中的模式定义片段是非标准化的。

7.5.1 元素<Request>

<Request>元素是 XACML 上下文模式定义中一个最高级的元素。<Request>元素是策略语言采用的一个抽象层。为了表达的简单性,本建议书根据对上下文的操作描述策略赋值。但是,不要求一个一致的 PDP 来实际地利用 XML 文档的形式来构建上下文示例。但是任何和 XACML 一致的系统都必须精确地生成相同的授权决定,就如同已经把所有的输入都转换成了一个<xacml-context:Request>元素的格式。

<Request>元素包含<Subject>、<Resource>、<Action>和<Environment>元素。可以存在多个<Subject>元素,而且在某些情况下,可能会存在多个<Resource>元素。每个子元素包含一个分别和主体、资源、动作和环境相关的<xacml-context:Attribute>元素序列。这些<Attribute>元素可以形成策略赋值的一部分。

```
<xs:element name="Request" type="xacml-context:RequestType"/>
<xs:complexType name="RequestType">
  <xs:sequence>
    <xs:element ref="xacml-context:Subject" maxOccurs="unbounded"/>
    <xs:element ref="xacml-context:Resource" maxOccurs="unbounded"/>
    <xs:element ref="xacml-context:Action"/>
    <xs:element ref="xacml-context:Environment"/>
  </xs:sequence>
</xs:complexType>
```


<Request>元素是 **RequestType** 复杂类型。

<Request>元素包含下述元素：

- <Subject> [1 个到多个]
通过列出和主体相关的一个<Attribute>元素序列规定关于一个请求上下文的主体的信息。允许 1 个或者多个<Subject>元素。一个主体是一个和接入请求相关联的实体。例如，一个主体可能表示发起应用的用户，通过这个应用发起了请求；另外一个主体可能表示应用的负责生成请求的可执行代码；另外一个主体可能表示正在执行应用的机器；而且另外一个主体可能表示作为资源的接收方的实体。必须把每个实体的属性封装到独立的<Subject>元素中。
- <Resource> [1 个到多个]
通过列出和资源相关的一个<Attribute>元素序列来指定请求访问的资源的信息。它可以包括一个<ResourceContent>元素。
- <Action> [必需]
通过列出和动作相关的一组<Attribute>元素定义了需要在资源上执行的请求的动作。
- <Environment> [必需]
包含了一组用于环境的<Attribute>属性。

7.5.2 元素<Subject>

<Subject>元素通过列出和主体相关的一个<Attribute>元素序列来定义一个主体。

```
<xs:element name="Subject" type="xacml-context:SubjectType"/>
<xs:complexType name="SubjectType">
  <xs:sequence>
    <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="SubjectCategory" type="xs:anyURI"
default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
</xs:complexType>
```

<Subject>元素是 **SubjectType** 复杂类型。

<Subject>元素包含下述元素和属性：

- SubjectCategory [可选]
本属性指示了父<Subject>在接入请求所扮演的角色。如果在一个给定的<Subject>元素中没有出现本属性，那么应该采用缺省值"urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"，它指示父<Subject>代表了最终负责发起接入请求的实体。
如果有多个<Subject>元素包含了一个"urn:oasis:names:tc:xacml:2.0:subject-category"，它们的属性值相同，那么 PDP 应该把这些元素的内容视为包含在同一个<Subject>元素里。
- <Attribute> [任何数量]
应用到主体的一个属性的序列。

典型地，一个<Subject>元素会包含一个<Attribute>，它的 AttributeId 为"urn:oasis:names:tc:xacml:1.0:subject:subject-id"，它包含了主体的身份。

一个<Subject>元素可以包含附加的<Attribute>元素。

7.5.3 元素<Resource>

<Resource>元素通过列出一个和资源相关的<Attribute>元素序列定义了请求要访问的资源的信息。它可以包含资源的内容。

```
<xs:element name="Resource" type="xacml-context:ResourceType"/>
<xs:complexType name="ResourceType">
  <xs:sequence>
    <xs:element ref="xacml-context:ResourceContent" minOccurs="0"/>
    <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<Resource>元素是 **ResourceType** 复杂类型。

<Resource>元素包含下列元素:

- <ResourceContent> [可选]
资源内容。
- <Attribute> [任意数量]
资源属性的一个序列。

<Resource>元素可以包含一个或者多个<Attribute>元素，它们的 AttributeId 的为 "urn:oasis:names:tc:xacml:2.0:resource:resource-id"。每个这样的<Attribute>应该是请求访问的单个资源身份的一个绝对的而且全解析的表示法。如果存在多个这样的绝对的而且全解析的表示法，那么如果定义了任何具有这个 AttributeId 的<Attribute>，那么应该规定用于资源身份的每个不同的表示法的<Attribute>。所有这样的<Attribute>元素应该代表用样的单个资源实例。一个用于特定资源的轮廓可以为资源实例定义一个单一的标准表示法；在这种情况下，任何具有这个 AttributeId 的<Attribute>都应该只采用这一个表示法。

<Resource>元素可以包含额外的<Attribute>元素。

7.5.4 元素<ResourceContent>

<ResourceContent>元素是一个用于资源内容的概念上的占位符。如果 XACML 策略通过一个<AttributeSelector>元素引用资源的内容，那么必须在 RequestContextPath 串中包括<ResourceContent>元素。

```
<xs:complexType name="ResourceContentType" mixed="true">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
```

<ResourceContent>元素是 **ResourceContentType** 复杂类型。

<ResourceContent>元素承认任何的元素和属性。

7.5.5 元素<Action>

<Action>元素通过列出一组和动作相关的<Attribute>元素定义了针对资源的请求的动作。

```
<xs:element name="Action" type="xacml-context:ActionType"/>
<xs:complexType name="ActionType">
  <xs:sequence>
    <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<Action>元素是 **ActionType** 复杂类型。

<Action>元素包含下列元素：

- <Attribute> [任意数量]
将要在资源上执行的动作属性列表。

7.5.6 元素<Environment>

<Environment>元素包含了环境的一组属性。

```
<xs:element name="Environment" type="xacml-context:EnvironmentType"/>
<xs:complexType name="EnvironmentType">
  <xs:sequence>
    <xs:element ref="xacml-context:Attribute"
minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<Environment>元素是 **EnvironmentType** 复杂类型。

<Environment>元素包含下列元素：

- <Attribute> [任意数量]
环境属性的一个列表。环境属性是和接入请求的资源、动作或者任何一个主体都不相关的属性。

7.5.7 元素<Attribute>

<Attribute>元素是请求上下文的一个主要摘要。它包含属性元数据和 1 个或者多个属性值。属性元数据包含属性标识符和属性发布者。策略中的<AttributeDesignator>和<AttributeSelector>元素可以通过这种元数据来引用属性。

```
<xs:element name="Attribute" type="xacml-context:AttributeType"/>
<xs:complexType name="AttributeType">
  <xs:sequence>
    <xs:element ref="xacml-context:AttributeValue" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
  <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
  <xs:attribute name="Issuer" type="xs:string" use="optional"/>
</xs:complexType>
```

<Attribute>元素是 **AttributeType** 复杂类型。

<Attribute>元素包含下述属性和元素：

- AttributeId [必需]
属性标识符。XACML 保留一定数量的标识符用来表示常用的属性。
- DataType [必需]
<xacml-context:AttributeValue>元素的内容的数据类型。它可以是本建议书定义的一个原类型也可以是<xacml-context>元素宣布的一个命名空间内定义的一个类型（初始的或者结构化的）。
- Issuer [可选]
属性发布者。例如，这个属性值可以是一个和一个公共密钥绑定的 x500Name，或者它可以是通过发布和可信赖团体在带外进行交换的其它标识符。
- <xacml-context:AttributeValue> [1 个到多个]
一个或者多个属性值。每个属性值都具有空的、出现一次或者出现多次的内容。

7.5.8 元素<AttributeValue>

<xacml-context:AttributeValue>元素包含一个属性的值。

```
<xs:element name="AttributeValue" type="xacml-context:AttributeValueType"/>
<xs:complexType name="AttributeValueType" mixed="true">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
```

<xacml-context:AttributeValue>元素是 **AttributeValueType** 复杂类型。

<xacml-context:AttributeValue>的数据类型应该通过使用父<Attribute>元素的 **DataType** 属性来定义。

7.5.9 元素<Response>

<Response>元素是 XACML 上下文模式定义中的一个最高级的元素。<Response>元素是策略语言使用的一个抽象层。任何采用 XACML 的私有系统都必须把一个 XACML 上下文<Response>元素转换为它的授权决定的格式。

<Response>元素封装 PDP 产生的授权决定。它包括一个或者多个结果的一个序列，每个被请求资源一个<Result>元素。某些执行情况可以返回多个结果，尤其是在为多个资源的请求 (Requests) 支持 XACML 轮廓的情况下。对多个结果的支持为可选。

```
<xs:element name="Response" type="xacml-context:ResponseType"/>
<xs:complexType name="ResponseType">
  <xs:sequence>
    <xs:element ref="xacml-context:Result" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<Response>元素是 **ResponseType** 复杂类型。

<Response>元素包含下述元素：

- <Result> [1 个到多个]
一个授权决定结果。

7.5.10 元素<Result>

<Result>元素代表用于通过 ResourceId 属性定义的资源的一个授权决定结果。它可以包含必须要由 PEP 完成的一组职责。如果 PEP 无法理解后者无法完成一个职责，那么它必须按照 PDP 已经拒绝访问被请求资源的情况来采取动作。

```
<xs:complexType name="ResultType">
  <xs:sequence>
    <xs:element ref="xacml-context:Decision"/>
    <xs:element ref="xacml-context:Status" minOccurs="0"/>
    <xs:element ref="xacml:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="ResourceId" type="xs:string" use="optional"/>
</xs:complexType>
```

<Result>元素是 **ResultType** 复杂类型。

<Result>元素包含下述属性和元素。

- ResourceId [可选]
被请求资源的标识符。如果这个属性被省略，那么通过相应的<Request>元素中的 "urn:oasis:names:tc:xacml:1.0:resource:resource-id" 资源属性定义了资源的身份。
- <Decision> [必需]
授权决定: "Permit"、"Deny"、"Indeterminate" 或者 "NotApplicable"。

- `<Status>` [可选]
指示在决定请求的赋值过程中是否出现了差错，而且还可以选择性地说明关于这些差错的信息。如果 `<Response>` 元素包含 `<Result>` 元素，而 `<Result>` 元素的 `<Status>` 元素都相同，而且 `<Response>` 元素包含在一个可以用来传递状态信息的协议封装中，那么可以把通用状态信息放在协议包装器中而且所有的 `<Result>` 元素都可以忽略这个 `<Status>` 元素。
- `<Obligations>` [可选]
必须要由 PEP 完成的一个职责列表。如果 PEP 无法理解或者无法完成职责，那么它必须按照 PDP 已经拒绝访问被请求的资源来采取动作。

7.5.11 元素 `<Decision>`

`<Decision>` 元素包含策略赋值的结果。

```
<xs:element name="Decision" type="xacml-context:DecisionType"/>
<xs:simpleType name="DecisionType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Permit"/>
    <xs:enumeration value="Deny"/>
    <xs:enumeration value="Indeterminate"/>
    <xs:enumeration value="NotApplicable"/>
  </xs:restriction>
</xs:simpleType>
```

`<Decision>` 元素是 **DecisionType** 简单类型。

`<Decision>` 元素的值具有下述含义：

- **Permit**: 允许请求的访问。
- **Deny**: 拒绝请求的访问。
- **Indeterminate**: PDP 无法赋值请求的接入。无法赋值的原因包括：缺少属性、在重新获得策略的过程中出现网络差错、在策略赋值的过程中出现用 0 作除数、决定请求或者策略中出现句法差错等。
- **NotApplicable**: PDP 没有任何可以应用到这个决定请求的策略。

7.5.12 元素 `<Status>`

`<Status>` 元素代表授权决定结果的状态。

```
<xs:element name="Status" type="xacml-context:StatusType"/>
<xs:complexType name="StatusType">
  <xs:sequence>
    <xs:element ref="xacml-context:StatusCode"/>
    <xs:element ref="xacml-context:StatusMessage" minOccurs="0"/>
    <xs:element ref="xacml-context:StatusDetail" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

`<Status>` 元素是 **StatusType** 复杂类型。

`<Status>` 元素包含下述元素：

- `<StatusCode>` [必需]
状态码。
- `<StatusMessage>` [可选]
描述状态码的一个状态消息。
- `<StatusDetail>` [可选]
附加的状态信息。

7.5.13 元素 `<StatusCode>`

`<StatusCode>` 元素包含了一个主要的状态码值和一个可选的次要的状态码。

```
<xs:element name="StatusCode" type="xacml-context:StatusCodeType"/>
<xs:complexType name="StatusCodeType">
  <xs:sequence>
    <xs:element ref="xacml-context:StatusCode" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="Value" type="xs:anyURI" use="required"/>
</xs:complexType>
```

<StatusCode>元素是 **StatusCodeType** 复杂类型。

<StatusCode>元素包含下述属性和元素：

- Value [必需]
B.8 示出了值的列表。
- <StatusCode> [任意数量]
次要的状态码。这个状态码限制了它的父状态码。

7.5.14 元素<StatusMessage>

<StatusMessage>元素是对状态码的自由格式的描述。

```
<xs:element name="StatusMessage" type="xs:string"/>
```

<StatusMessage>元素是 **xs:string** 类型。

7.5.15 元素<StatusDetail>

<StatusDetail>元素限制了带有额外信息的<Status>元素。

```
<xs:element name="StatusDetail" type="xacml-context:StatusDetailType"/>
<xs:complexType name="StatusDetailType">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

<StatusDetail>元素是 **StatusDetailType** 复杂类型。

<StatusDetail>元素允许任意的 XML 内容。

包含一个<StatusDetail>元素是可选的。但是，如果一个 PDP 返回一个下述由 XACML 定义的<StatusCode>值而且包括一个<StatusDetail>元素，那么应用下面的规则。

```
urn:oasis:names:tc:xacml:1.0:status:ok
```

在"ok"状态值相组合的情况下，一个 PDP 不能返回一个<StatusDetail>元素。

```
urn:oasis:names:tc:xacml:1.0:status:missing-attribute
```

一个 PDP 可以选择不返回任何<StatusDetail>信息或者可以选择返回一个包含一个或者多个<xacml-context:MissingAttributeDetail>元素的<StatusDetail>元素。

```
urn:oasis:names:tc:xacml:1.0:status:syntax-error
```

在和"syntax-error"状态值组合的情况下，一个 PDP 不能返回一个<StatusDetail>元素。一个句法错误代表可能正在使用的策略中出现了一个问题或者是请求上下文出现了问题。PDP 可以返回一个描述问题的<StatusMessage>。

```
urn:oasis:names:tc:xacml:1.0:status:processing-error
```

在和"processing-error"状态值相组合情况下，一个 PDP 不能返回<StatusDetail>元素。这个状态码表示 PDP 中的一个内部问题。出于安全原因，PDP 可以选择不给 PEP 返回更多的信息。如果出现了用 0 作除数差错或者其它的计算问题，PDP 可以返回一个描述差错种类的<StatusMessage>。

7.5.16 元素<MissingAttributeDetail>

<MissingAttributeDetail>元素传递策略赋值需要的请求上下文中缺少的属性信息。

```
<xs:element name="MissingAttributeDetail"
type="xacml-context:MissingAttributeDetailType"/>
<xs:complexType name="MissingAttributeDetailType">
<xs:sequence>
<xs:element ref="xacml-context:AttributeValue" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
<xs:attribute name="DataType" type="xs:anyURI" use="required"/>
  <xs:attribute name="Issuer" type="xs:string" use="optional"/>
</xs:complexType>
```

<MissingAttributeDetail>元素是 **MissingAttributeDetailType** 复杂类型。

<MissingAttributeDetail>元素包含下述属性和元素：

- AttributeValue [可选]
缺少的属性的必需值。
- <AttributeId> [必需]
缺少的属性的标识符。
- <DataType> [必需]
缺少的属性的数据类型。
- Issuer [可选]
如果提供了这个属性，那么它应该规定了缺少的属性的必需的 Issuer。

如果 PDP 在<MissingAttributeDetail>元素中包含了<xacml-context:AttributeValue>元素，那么它这是该属性可接受的值。如果没有包括<xacml-context:AttributeValue>元素，那么它指示在赋值过程中 PDP 没有成功解析的属性的名字。属性列表可以是部分的也可以是全部的。PDP 不保证提供的缺少的值或者属性可以满足策略的需要。

7.6 XACML功能需求

本部分定义了某些和一个特定的 XACML 元素的成果或者使用没有直接关联的功能需求。

7.6.1 策略执行点

本部分描述了对 PEP 的需求。

如果 PEP 掌管着一个资源组的访问而且需要向 PDP 要求一个授权决定，那么应用运作在 PEP 角色之中。PEP 必需遵守 PDP 授权决定。

7.6.1.1 基础 PEP

如果决定为"Permit"，那么 PEP 应该允许访问。如果决定带有一定的职责，那么只有在 PEP 理解而且能够履行这些职责时，它允许访问。

如果决定为"Deny"，那么 PEP 应该拒绝访问。如果决定带有一定的职责，那么只有 PEP 理解而且能够履行这些职责时，它拒绝访问。

如果决定为"Not Applicable"，那么没有定义 PEP 的动作。

如果决定为"Indeterminate"，那么没有定义 PEP 的动作。

7.6.1.2 偏拒绝 PEP

如果决定为"Permit"，那么 PEP 应该允许访问。如果决定带有一定的职责，那么只有 PEP 理解而且它能够履行那些职责时，它允许访问。

所有其它的决定的结果都应导致拒绝访问。

注 — 没有禁止其它动作，例如咨询其它的PDP，决定请求的再形成/再服从等。

7.6.1.3 偏允许 PEP

如果决定是"Deny"，那么 PEP 应该拒绝访问。如果决定带有一定的职责，那么只有 PEP 理解而且它应该能够履行这些职责时，它拒绝访问。

所有其它的决定的结果都应导致允许访问。

注 — 没有禁止其它的动作，例如咨询其它的PDP，决定请求的再形成/再服从等。

7.6.2 属性赋值

通过上下文处理器在请求上下文中描述属性，而无需考虑它们是否出现在初始的决定请求中，而且它们还通过主体、资源、动作和环境属性指示器和属性选择者来在策略中引用。一个指定的属性是特定的主体、资源、动作和环境属性指示器和属性选择者所采用的分别用来引用请求上下文中的主体、资源、动作和环境元素的属性的准则的术语。

7.6.2.1 结构化属性

<xacml:AttributeValue>和<xacml-context:AttributeValue>元素可以包含一个结构化 XML 数据类型的实例，例如<ds:KeyInfo>。本建议书支持几种对比这种元素内容的方式。

- 1) 如下面描述，在某些情况下，这些元素可以利用一个XACML功能串对这些元素进行比较，例如"string-regexp-match"。这就要求提交给元素的数据类型为"http://www.w3.org/2001/XMLSchema#string"。例如，一个实际上是ds:KeyInfo/KeyName的结构化数据类型会以下面的形式出现在上下文中：

```
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
  <ds:KeyName>jhibbert-key</ds:KeyName>
</AttributeValue>
```

通常，这种方式只有在结构化数据类型相当简单的情况下才适用。

- 2) 可以利用一个<AttributeSelector>元素通过一个XPath表达式的方法来选择一個结构化数据类型的一个叶元素的内容。然后通过所支持的适用于这个原数据类型的XACML功能对这个值进行比较。这个方法要求PDP支持可选XPath表达式特性。
- 3) 可以利用一个<AttributeSelector>元素通过一个XPath表达式来在结构化数据类型中选择一个节点。然后可以使用A.3中描述的一个基于XPath的功能来对节点进行比较。这种方式要求PDP支持可选XPath表达式和XPath功能特性。

7.6.2.2 属性包

XACML 定义了它的数据类型的隐式集合。XACML 把每个单一数据类型的值的集合称作一个包。需要数据包的主要原因是一个 XML 资源或者 XACML 请求上下文选取节点可能会返回多个值。

<AttributeSelector>元素使用一个 XPath 表达式来定义从一个 XML 资源对数据的选取。把一个 XPath 表达式的结果称为节点集，它包含了来自 XML 资源并且和 XPath 表达式中的判定相匹配的所有的叶节点。根据 W3C XPath:1999 中提供的各种索引功能，它应该得到暗示，就是说得到的节点集是匹配节点的一个集合。本建议书还定义了<AttributeDesignator>元素在对待 XACML 请求上下文中的属性的时候应该具有相同的匹配方法。

包中的值是无顺序的，而且有些值可以存在副本。应该不存在一个包再包含包的概念，或者说一个包包含不同类型的值（也就是说，XACML 中的一个包应该只包含相同数据类型的值）。

7.6.2.3 多值属性

如果一个请求上下文中的单一的<Attribute>元素包含多个<xacml-context:AttributeValue>子元素，那么通过赋值<Attribute>元素得到的一系列值必须和通过赋值一个上下文得到的一系列值相同，在这个上下文中，每个<xacml-context:AttributeValue>元素出现在独立的<Attribute>元素中，它们携带相同元数据。

7.6.2.4 属性匹配

一个指定的属性包括特定的准则，通过这些准则来匹配上下文中的属性。一个属性定义了一个 `AttributeId` 和 `DataType`，同时一个指定的属性还定义了 `Issuer`。如果一个指定的属性和一个属性的各自的 `AttributeId`、`DataType` 和可选的 `Issuer` 属性在它们特定的上下文的元素—主体、资源、动作或者环境里相匹配，那么这两个元素应该相匹配。指定属性的 `AttributeId` 必须通过 URI 等同性匹配相应上下文属性中的 `AttributeId`。指定属性的 `DataType` 必须通过 URI 等同性匹配相应上下文属性中的 `DataType`。如果指定属性提供了 `Issuer`，那么它必须使用 `urn:oasis:names:tc:xacml:1.0:function:string-equal` 功能来匹配相应上下文属性中的 `Issuer`。如果在指定的属性中没有提供 `Issuer`，那么应该通过 `AttributeId` 和 `DataType` 来独自掌管上下文属性和指定属性的匹配，而无需考虑在相应的上下文属性中是否出现了 `Issuer` 或者实际的 `Issuer` 值。在属性选择者情况下，属性和指定属性的匹配应该通过 XPath 表达式和 `DataType` 来管理。

7.6.2.5 属性检索

PDP 应该从上下文处理器处请求请求上下文中属性的值。PDP 应该引用属性就如同它们在一个物理上下文中，但是上下文处理器负责通过任何它认为适当的方式获得并且提供请求的值。上下文处理器应该返回和属性指定者或者属性选择者相匹配的属性的值并且把它们组合到一个具有特定的数据类型的一系列值中。如果没有来自请求上下文中的属性相匹配，那么应该认为属性丢失。如果属性丢失，那么 `MustBePresent` 管理属性指定者或者属性选择者是否返回了一个空包或者一个 "Indeterminate" 结果。如果 `MustBePresent` 为 "False" (缺省值)，那么一个缺少的属性应该得到一个空包。如果 `MustBePresent` 为 "True"，那么一个缺少的属性应该得到 "Indeterminate"。应该根据所包含的表达式、规则、策略和策略集的规范来处理 "Indeterminate" 结果。如果结果为 "Indeterminate"，那么应该在授权决定中列出属性的 `AttributeId`、`DataType` 和 `Issuer`。但是，出于安全的原因，一个 PDP 可以选择不返回这样的信息。

7.6.2.6 环境属性

如果在决定请求中为这些属性中的一个提供了值，那么上下文处理器应该利用这个值。否则，上下文处理器应该提供一个值。在日期和时间属性情况下，提供的值应该具有“应用到决定请求的日期和时间”的语义。

7.6.3 表达式赋值

XACML 根据下面列出的元素定义了表达式，其中 `<Apply>` 和 `<Condition>` 元素递归式地组成了更大的表达式。有效的表达式类型应该正确，它意味着包含在 `<Apply>` 和 `<Condition>` 元素中的每个元素的类型应该和各自通过 `FunctionId` 属性命名的函数的参数类型一致。通过 `<Apply>` 或者 `<Condition>` 元素得到的类型应该是通过函数而得到的类型，这个类型对于一个原数据类型或者原数据类型的一个包来讲，范围变窄了，但是类型统一了。XACML 定义了一个针对 "Indeterminate" 的赋值结果，"Indeterminate" 被认为是一个无效表达式的结果，或者是在赋值一个表达式的过程中出现的操作错误。

XACML 定义了 `<Expression>` 元素的替代组中的元素：

- `<xacml:AttributeValue>`
- `<xacml:SubjectAttributeDesignator>`
- `<xacml:ResourceAttributeDesignator>`
- `<xacml:ActionAttributeDesignator>`
- `<xacml:EnvironmentAttributeDesignator>`
- `<xacml:AttributeSelector>`
- `<xacml:Apply>`
- `<xacml:Condition>`
- `<xacml:Function>`
- `<xacml:VariableReference>`

7.6.4 算法赋值

IEEE 754 定义了如何赋值一个上下文中的算术函数，它为精度，舍入等定义了缺省的情况。XACML 应该根据通过双精度而增强了的扩展缺省上下文来赋值所有的整数和双函数。

- 标志：全部置0；
- 限制启用：除了在"division-by-zero"（用0作除数）限制启用的情况下，应该置为1外，都置0；
- 精度：设置为指定的双精度；
- 舍入：设置为四舍五入。

7.6.5 匹配赋值

属性匹配元素出现在规则、策略和策略集的<Target>中，如下所示：

- <SubjectMatch>
- <ResourceMatch>
- <ActionMatch>
- <EnvironmentMatch>

这些元素分别代表用于主体、资源、动作和环境的属性的布尔表达式。一个匹配元素包含了一个 MatchId 属性，该属性定义了在执行匹配赋值过程中将要使用的函数，定义了上下文中属性的一个 <xacml:AttributeValue> 和一个 <AttributeDesignator> 或者 <AttributeSelector> 元素需要和定义的值相匹配。

MatchId 属性应该规定一个对比两个变量的函数，返回一个"http://www.w3.org/2001/XMLSchema#boolean" 结果类型。把在匹配元素中定义的属性值作为 MatchId 函数的第一个变量提供给它。应该把由 <AttributeDesignator> 或者 <AttributeSelector> 元素返回的一系列元素作为 MatchId 函数的第二个变量提供给它，解释如下。<xacml:AttributeValue> 的 DataType 应该匹配 MatchId 函数所期望的第一个变量的数据类型。<AttributeDesignator> 或者 <AttributeSelector> 元素的 DataType 应该匹配 MatchId 函数所期望的第二个变量的数据类型。

满足作为一个 MatchId 属性值来使用的需求的 XACML 标准函数为：

```
urn:oasis:names:tc:xacml:2.0:function:-type-equal
urn:oasis:names:tc:xacml:2.0:function:-type-greater-than
urn:oasis:names:tc:xacml:2.0:function:-type-greater-than-or-equal
urn:oasis:names:tc:xacml:2.0:function:-type-less-than
urn:oasis:names:tc:xacml:2.0:function:-type-less-than-or-equal
urn:oasis:names:tc:xacml:2.0:function:-type-match
```

除此以外，严格地位于 XACML 扩展内的函数可以作为 MatchId 属性的一个值出现，而且只要这些扩展函数返回一个布尔结果并且采用两个单独的基类作为它的输入，那么这些函数可以采用同样为扩展的数据类型。作为 MatchId 属性值的函数应该很容易索引。采用无法索引或者复杂的函数可能会阻碍决定请求的有效赋值。

用于一个匹配元素的赋值语义如下所示。如果在赋值 <AttributeDesignator> 或者 <AttributeSelector> 元素过程中，出现了运算差错，那么整个表达式的结果应该是"Indeterminate"。如果 <AttributeDesignator> 或者 <AttributeSelector> 元素用来赋值一个空包，那么表达式的结果应该是"False"。否则，应该把 MatchId 函数应用在<xacml:AttributeValue>和从<AttributeDesignator>或者 <AttributeSelector> 元素返回的组中的每个元素之间。如果至少有一个这样的函数应用赋值为"True"，那么整个表达式的结果应该为"True"。否则如果至少有一个函数应用结果为"Indeterminate"，整个结果为"Indeterminate"。最后，如果所有的函数应用赋值为"False"，那么整个表达式的结果应该为"False"。

在某种情况下表达一个目标匹配元素的语义也是可能的。例如，比较一个以名字"John"开头的"subject-name"的目标匹配表达式可以表示如下：

```
<SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    John.*
  </AttributeValue>
```

```

<SubjectAttributeDesignator
  AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
  DataType="http://www.w3.org/2001/XMLSchema#string"/>
</SubjectMatch>

```

可选地，同样的匹配语义可以通过"urn:oasis:names:tc:xacml:1.0:function:any-of"函数表达为某种条件下的一个<Apply>元素，如下所示：

```

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
  <Function
    FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match"/>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    John.*
  </AttributeValue>
  <SubjectAttributeDesignator
    AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
    DataType="http://www.w3.org/2001/XMLSchema#string"/>
</Apply>

```

7.6.6 目标赋值

如果目标中规定的主体、资源、动作和环境都和请求上下文的值匹配，那么目标值应该为"Match"。如果目标中规定的主体、资源、动作和环境，有任何一个是"Indeterminate"，那么目标都应该是"Indeterminate"。否则，目标应该为"No match"。目标匹配表如表 7-1 所示。

表 7-1/X.1142—目标匹配表

主体值	资源值	动作值	环境值	目标值
"Match"	"Match"	"Match"	"Match"	"Match"
"No match"	"Match"或"No match"	"Match"或"No match"	"Match"或"No match"	"No match"
"Match" 或 "No match"	"No match"	"Match"或"No match"	"Match"或"No match"	"No match"
"Match"或"No match"	"Match"或"No match"	"No match"	"Match"或"No match"	"No match"
"Match"或"No match"	"Match"或"No match"	"Match"或"No match"	"No match"	"No match"
"Indeterminate"	随意	随意	随意	"Indeterminate"
随意	"Indeterminate"	随意	随意	"Indeterminate"
随意	随意	"Indeterminate"	随意	"Indeterminate"
随意	随意	随意	"Indeterminate"	"Indeterminate"

如果主体、资源、动作和环境分别至少有一个<Subject>、<Resource>、<Action>或者<Environment>元素和请求上下文中的一个值匹配，那么主体、资源、动作和环境应该匹配请求上下文中的值。主体匹配表参见表 7-2。资源、动作和环境匹配表与之类似。

表 7-2/X.1142—主体匹配表

<Subject> 值	<Subjects>值
至少一个 "Match"	"Match"
没有 matches 并且至少一个 "Indeterminate"	"Indeterminate"
所有都是 "No match"	"No match"

如果主体、资源、动作或者环境的各自的<SubjectMatch>、<ResourceMatch>、<ActionMatch> 或者<EnvironmentMatch>元素的值都为"True"，那么一个主体、资源、动作或者环境应该和请求上下文中的一个值匹配。

主体匹配表如表 7-3 所示。资源、动作和环境匹配表类似。

表 7-3/X.1142—主体匹配表

<SubjectMatch> 值	<Subject> 值
所有都为"True"	"Match"
没有"False"而且至少有一个为"Indeterminate"	"Indeterminate"
至少一个为"False"	"No match"

7.6.7 VariableReference赋值

<VariableReference> 元素引用了一个单一的包含在同一个 <Policy> 元素中的 <VariableDefinition> 元素。把没有引用包含在一个封闭 <Policy> 元素中一个特定 <VariableDefinition>元素的<VariableReference>称作是未定义引用。带有未定义引用的策略是无效的。

在出现一个<VariableReference>的地方，它的效果就好像在<VariableDefinition>元素中定义的 <Expression>元素的文本替代了<VariableReference>元素。任何保留这种语义的赋值方案都可以接受。例如，可以把<VariableDefinition>元素中的表达式赋值为一个特定的值并且为多个引用进行存储而无需因果关系（也就是说，在整个策略赋值过程中，一个<Expression>元素的值保持不变。）。这个特性是 XACML 作为一种陈述性语言的一个好处。

7.6.8 条件赋值

如果缺少<Condition>元素或者<Condition>元素赋值为"True"，那么条件值应该为"True"。如果 <Condition>元素赋值为"False"，那么它的值应该为"False"。如果包含在<Condition>元素中的表达式赋值为 "Indeterminate"，那么条件值应该为"Indeterminate"。

7.6.9 规则赋值

一个规则具有一个可以通过赋值它的内容而计算出来的值。规则赋值涵盖了独立的规则的目标赋值和条件赋值。规则真值表如表 7-4 所示。

表 7-4/X.1142—规则真值表

目 标	条 件	规 则 值
"Match"	"True"	Effect
"Match"	"False"	"NotApplicable"
"Match"	"Indeterminate"	"Indeterminate"
"No-match"	随意	"NotApplicable"
"Indeterminate"	随意	"Indeterminate"

如果目标值为"No-match"或者"Indeterminate"，那么规则值应该分别是"NotApplicable"或者 "Indeterminate"，无需考虑条件值。因此，在这些情况下，不需要赋值条件。

如果目标是"Match"，条件值是"True"，那么在封闭元素<Rule>中定义的结果应该决定规则的值。

7.6.10 策略赋值

一个策略值应该只由它的内容，同时考虑和请求上下文的内容的关系而决定。应该通过对策略目标和规则的赋值来决定一个策略值。

对一个策略的目标进行赋值从而决定策略的应用性。如果目标赋值为"Match"，那么应该通过已定义的规则组合算法对策略的规则进行赋值来决定策略的值。如果目标赋值为"No-match"，那么策略值应该为"NotApplicable"。如果目标赋值为"Indeterminate"，那么策略值应该为"Indeterminate"。

策略真值表如表 7-5 所示。

表 7-5/X.1142—策略真值表

目 标	规 则 值	策 略 值
"Match"	至少一个规则值是它的结果	由规则组合算法确定
"Match"	所有规则值都是"NotApplicable"	"NotApplicable"
"Match"	至少一个规则值是 "Indeterminate"	由规则组合算法确定
"No-match"	随意	"NotApplicable"
"Indeterminate"	随意	"Indeterminate"

规则值是"至少一个规则值是它的结果",意味着可以是缺少<Rule>元素,或者是包含在策略中的一个或者多个规则适用于决定请求(也就是说,它返回的值为"Effect")。如果包含在策略中的规则都不适用于请求而且如果包含在策略中的规则都没有返回一个"Indeterminate"值,那么应该采用“所有规则值都是 'NotApplicable'”这个规则值。如果包含在策略中的规则都不适用于请求,但是一个或者多个规则返回了"Indeterminate"值,那么规则应该赋值为"至少一个规则值是 'Indeterminate'"。

如果目标值是"No-match"或者"Indeterminate",那么策略值应该分别是"NotApplicable"或"Indeterminate",而无需考虑规则的值。因此,在这些情况下,不需要对规则进行赋值。

如果目标值是"Match" 而且规则值为"至少一个规则值是它的结果"或者"至少一个规则值是 'Indeterminate'",那么策略中定义的规则组合算法应该决定策略值。

注意在本建议书中定义的规则组合算法都不需要参数。但是,非标准的组合算法可能会需要参数。在这样的情况下,在赋值策略的时候,必须考虑和规则相关的这些参数值。在组合算法的说明中需要定义参数和它们的类型。如果执行情况支持组合参数而且如果策略中出现了组合参数,那么必须把参数值提供给组合算法执行情况。

7.6.11 策略集赋值

策略集的值应该由它的内容以及考虑到和请求上下文的关系来决定。应该根据已经定义的策略组合算法通过赋值策略集的目标、策略和策略集来决定一个策略集的值。

应该赋值策略集的目标来决定策略集的可应用性。如果目标赋值为"Match",那么应该根据已经定义的策略组合算法,通过赋值策略集的策略和策略集来决定策略集的值。如果目标赋值为"No-match",那么策略集的值应该为"NotApplicable"。如果目标赋值为"Indeterminate",那么策略集的值应该是"Indeterminate"。

策略集真值表如表 7-6 所示。

表 7-6/X.1142—策略集真值表

目 标	策 略 值	策 略 组 值
"Match"	至少一个策略值是它的决定	由策略组合算法确定
"Match"	所有策略值都是 "NotApplicable"	"NotApplicable"
"Match"	至少一个策略值是 "Indeterminate"	由策略组合算法确定
"No-match"	随意	"NotApplicable"
"Indeterminate"	随意	"Indeterminate"

如果策略集包含的或者引用的策略或者策略集都不适用于决定请求,或者有一个或者多个所包含的或者引用的策略或者策略集适用于决定请求,那么应该利用策略值"至少一个策略值是它的决定"(也就是说,返回由它的组合算法决定的一个值)。如果策略集包含的或者引用的策略或者策略集都不适用于决定请求而且如果策略集包含的或者引用的策略或者策略集都没有返回一个"Indeterminate"值,那么应该采用策略值"所有策略值都是 'NotApplicable'"。如果策略集包含的或者引用的策略或者策略集都不适用于请求但是有一个或者多个策略或者策略集返回了一个"Indeterminate"值,那么策略应该赋值为"至少一个策略值是 'Indeterminate'"。

如果目标值为"No-match"或者"Indeterminate",那么策略集值应该分别是"NotApplicable"或者 "Indeterminate",无需考虑策略的值。因此,在这些情况下,不需要赋值策略。

如果目标值为"Match"而且策略值为"至少一个策略值是它的决定"或者"至少一个策略值是 'Indeterminate'",那么应该由在策略集中定义的策略组合算法决定策略集的值。

注意由 XACML 2.0 定义的策略组合算法都不需要参数。但是,有些非标准的组合算法可能会用到参数。在这种情况下,在赋值策略集的时候,必须考虑这些和策略相关联的参数值。在组合算法的说明中应该规定参数和它们的类型。如果执行情况支持组合参数而且如果在策略中出现了组合参数,那么必须把参数值提供给组合算法执行情况。

7.6.12 层次资源

一个资源通常会按照层次的结构来管理(例如,文件系统,XML 文档)。XACML 为支持层次资源提供了几个可选的机制,参见本建议书的讨论。

7.6.13 授权决定

关于一个特定的决定请求,由一个策略组合算法和一组策略和/或者策略集定义了 PDP。PDP 应该返回一个应答上下文,就如同它已经赋值了一个由这个策略组合算法和这组策略和/或者策略集组成的策略集。

PDP 必须按照 7.4 和 7.6 中的规定来赋值策略集。PDP 必须返回一个带有一个值为"Permit"、"Deny"、"Indeterminate"或者"NotApplicable"的<Decision>元素的应答上下文。

如果 PDP 无法作出决定,那么应该返回一个"Indeterminate" <Decision>元素。

7.6.14 职责

一个策略或者策略集可以包含一个或者多个职责。在赋值这样一个策略或者策略集的时候,如果被赋值的策略或者策略集的结果和职责的 FulfillOn 属性值相匹配,那么应该把职责传递给赋值的下一级(封闭的或者引用的策略、策略集或者授权决定)。

作为这个流程的一个结果,如果没有赋值从 PEP 得到的策略或者策略集,或者如果它们的赋值结果为"Indeterminate"或"NotApplicable",或者如果赋值策略或者策略集的决定结果和赋值一个封闭策略集的决定结果不匹配,那么不应该向 PEP 返回一个职责。

如果把 PDP 的赋值看作是策略集或者策略的一个树,每个都返回"Permit"或者"Deny",那么 PDP 向 PEP 返回的职责集应该只包括和这些路径相关联的职责,在这些路径处,每个赋值级别的结果都和 PDP 正在返回的结果相同。在缺少决定无法接受的情况下,应该采用一个确定的组合算法,例如 ordered-deny-override 算法。

7.6.15 例外处理

XACML 定义了在下述情况下 PDP 的动作。

7.6.15.1 不支持的泛函性

如果 PDP 试图去赋值一个包含了一个可选的 PDP 不支持的元素类型或者函数的策略集或者策略,那么 PDP 应该返回一个"Indeterminate"的<Decision>值。如果也返回了一个<StatusCode>元素,那么在不支持的元素类型的情况下,它的值应该是"urn:oasis:names:tc:xacml:1.0:status:syntax-error",在不支持的函数的情况下,它的值应该是"urn:oasis:names:tc:xacml:1.0:status:processing-error"。

7.6.15.2 句法和类型差错

如果在接收到一个决定请求的时候,XACML PDP 赋值一个包含无效句法的策略,那么策略的结果应该是"Indeterminate",它带有的 StatusCode 值为:

```
"urn:oasis:names:tc:xacml:1.0:status:syntax-error"
```

如果在接收到一个决定请求的时候，XACML PDP 赋值一个包含无效静态数据类型的策略，那么策略的结果应该是"Indeterminate"，它带有的 StatusCode 值为：

```
"urn:oasis:names:tc:xacml:1.0:status:processing-error"
```

7.6.15.3 缺少属性

如果在请求上下文中缺少了用于一个策略中的任何属性指定者或者选择起的匹配属性，那么应该得到一个包含"Indeterminate"值的<Decision>元素。如果在这种情况下，提供了状态码，那么应该采用值：

```
"urn:oasis:names:tc:xacml:1.0:status:missing-attribute"
```

来指示为了获得一个权威的决定，需要更多的信息。在这种情况下，<Status>元素可能会列出 PDP 精炼决定所需要的主体、资源、动作或环境的任何属性的名字和数据类型。一个 PEP 可以重新提交一个精确的请求上下文，通过为在前面的应答中列出的属性名字增加属性值来应答一个内容为"Indeterminate"的<Decision>元素，它带有的一个状态码是：

```
"urn:oasis:names:tc:xacml:1.0:missing-attribute"
```

如果 PDP 返回了一个内容为"Indeterminate"的<Decision>元素，它带有的状态码为：

```
"urn:oasis:names:tc:xacml:1.0:missing-attribute"
```

由于已经在初始的请求中列出了它们的值，它不再列出主体、资源、动作或者环境的任何属性的名字和数据类型。注意，这个需求强制 PDP 最终返回一个带有其它状态码的值为"Permit"、"Deny"或者"Indeterminate"的授权决定，以应答连续精确的请求。

7.7 XACML扩展点

本部分描述了 XACML 模型和模式定义中可以增加扩展的点，本部分为资料性的。

7.7.1 可扩展的XML属性类型

下述 XML 属性具有 URI 值。它们可以通过生成和用于这些属性的新语义相关联的新 URI 来进行扩展。

- AttributeId;
- DataType;
- FunctionId;
- MatchId;
- ObligationId;
- PolicyCombiningAlgId;
- RuleCombiningAlgId;
- StatusCode;
- SubjectCategory.

7.7.2 结构化属性

<xacml:AttributeValue>和<xacml-context:AttributeValue>元素可以包含一个结构化 XML 数据类型的实例。下面列出了需要 XACML 扩展的一些技术。

- 1) 对于一个给定的结构化数据类型，XACML用户的一个团体可以为具有和一个XACML定义的原数据类型一致的结构化数据类型的叶子元素定义新的属性标识符。利用这些新的属性标识符，PEP或者用户的团体所采用的上下文处理器可以把结构化数据类型实例变成一个独立<Attribute>元素的序列。可以通过XACML定义的功能对每个这样的<Attribute>元素进行比较。通过这种方式，结构化数据类型本身永远不会出现在一个<xacml-context:AttributeValue>元素中。

- 2) XACML用户的一个团体可以定义一个新的功能，通过该功能进行一个结构化数据类型值与其它值之间的比较。可能只有支持新功能的PDP可以利用该方式。

7.8 一致性

XACML 定义了一些具有一些特殊应用的功能。因此，不要求声明和该建议书一致的执行情况支持这些功能。

本部分列出了本建议书中的必须包括在声明与 XACML v2.0 一致的 PDP 执行情况中的部分。

注 — 对执行情况来讲，“M”为必备的，“O”为可选。

7.8.1 模式定义元素

执行情况必须支持这些标记为“M”的模式定义元素。

元素名称	M/O
xacml-context:Action	M
xacml-context:Attribute	M
xacml-context:AttributeValue	M
xacml-context:Decision	M
xacml-context:Environment	M
xacml-context:MissingAttributeDetail	M
xacml-context:Obligations	O
xacml-context:Request	M
xacml-context:Resource	M
xacml-context:ResourceContent	O
xacml-context:Response	M
xacml-context:Result	M
xacml-context:Status	M
xacml-context:StatusCode	M
xacml-context:StatusDetail	O
xacml-context:StatusMessage	O
xacml-context:Subject	M
xacml:Action	M
xacml:ActionAttributeDesignator	M
xacml:ActionMatch	M
xacml:Actions	M
xacml:Apply	M
xacml:AttributeAssignment	O
xacml:AttributeSelector	O
xacml:AttributeValue	M
xacml:CombinerParameters	O
xacml:CombinerParameter	O
xacml:Condition	M
xacml:Description	M
xacml:Environment	M
xacml:EnvironmentMatch	M
xacml:EnvironmentAttributeDesignator	M
xacml:Environments	M
xacml:Expression	M
xacml:Function	M
xacml:Obligation	O
xacml:Obligations	O
xacml:Policy	M
xacml:PolicyCombinerParameters	O

元素名称	M/O
xacml:PolicyDefaults	O
xacml:PolicyIdReference	M
xacml:PolicySet	M
xacml:PolicySetDefaults	O
xacml:PolicySetIdReference	M
xacml:Resource	M
xacml:ResourceAttributeDesignator	M
xacml:ResourceMatch	M
xacml:Resources	M
xacml:Rule	M
xacml:RuleCombinerParameters	O
xacml:Subject	M
xacml:SubjectMatch	M
xacml:Subjects	M
xacml:Target	M
xacml:VariableDefinition	M
xacml:VariableReference	M
xacml:XpathVersion	O

7.8.2 标识符前缀

由 XACML 保留下面的标识符前缀。

标识符
urn:oasis:names:tc:xacml:2.0
urn:oasis:names:tc:xacml:2.0:conformance-test
urn:oasis:names:tc:xacml:2.0:context
urn:oasis:names:tc:xacml:2.0:example
urn:oasis:names:tc:xacml:1.0:function
urn:oasis:names:tc:xacml:2.0:function
urn:oasis:names:tc:xacml:2.0:policy
urn:oasis:names:tc:xacml:1.0:subject
urn:oasis:names:tc:xacml:1.0:resource
urn:oasis:names:tc:xacml:1.0:action
urn:oasis:names:tc:xacml:1.0:environment
urn:oasis:names:tc:xacml:1.0:status

7.8.3 算法

执行情况必须包括和下面标着“M”的标识符相关联的规则和策略组合算法。

算法	M/O
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides	M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides	M
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:Permit-overrides	M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:Permit-overrides	M
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable	M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable	M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable	M
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides	M
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides	M
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-Permit-overrides	M
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-Permit-overrides	M

7.8.4 状态码

执行情况对<StatusCode>元素的支持为可选，但是如果支持<StatusCode>元素，那么必须支持下面的状态码而且必须通过 XACML 已经定义的方式来使用它们。

标识符	M/O
urn:oasis:names:tc:xacml:1.0:status:missing-attribute	M
urn:oasis:names:tc:xacml:1.0:status:ok	M
urn:oasis:names:tc:xacml:1.0:status:processing-error	M
urn:oasis:names:tc:xacml:1.0:status:syntax-error	M

7.8.5 属性

执行情况必须支持和下面的 XACML 定义的标识符相关联的属性。如果在决定请求中没有出现这些属性的值，那么必须由上下文处理器来提供这些值。因此，和大多数其他属性不同，它们的语义对 PDP 不透明。

标识符	M/O
urn:oasis:names:tc:xacml:1.0:environment:current-time	M
urn:oasis:names:tc:xacml:1.0:environment:current-date	M
urn:oasis:names:tc:xacml:1.0:environment:current-dateTime	M

7.8.6 标识符

执行情况必须通过 XACML 已经定义的方法来使用和下面的标识符相关联的属性。因为这些属性的语义对 PDP 来讲是透明的，因此这个需求主要适用于使用 XACML 的 PAP 或者 PEP 的执行情况。

标识符	M/O
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name	O
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address	O
urn:oasis:names:tc:xacml:1.0:subject:authentication-method	O
urn:oasis:names:tc:xacml:1.0:subject:authentication-time	O
urn:oasis:names:tc:xacml:1.0:subject:key-info	O
urn:oasis:names:tc:xacml:1.0:subject:request-time	O
urn:oasis:names:tc:xacml:1.0:subject:session-start-time	O
urn:oasis:names:tc:xacml:1.0:subject:subject-id	O
urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier	O
urn:oasis:names:tc:xacml:1.0:subject-category:access-subject	M
urn:oasis:names:tc:xacml:1.0:subject-category:codebase	O
urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject	O
urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject	O
urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine	O
urn:oasis:names:tc:xacml:1.0:resource:resource-location	O
urn:oasis:names:tc:xacml:1.0:resource:resource-id	M
urn:oasis:names:tc:xacml:1.0:resource:simple-file-name	O
urn:oasis:names:tc:xacml:1.0:action:action-id	O
urn:oasis:names:tc:xacml:1.0:action:implied-action	O

7.8.7 数据类型

执行情况必须支持和下面标记着“M”的标识符相关联的数据类型。

数据类型	M/O
http://www.w3.org/2001/XMLSchema#string	M
http://www.w3.org/2001/XMLSchema#boolean	M
http://www.w3.org/2001/XMLSchema#integer	M
http://www.w3.org/2001/XMLSchema#double	M

数据类型	M/O
http://www.w3.org/2001/XMLSchema#time	M
http://www.w3.org/2001/XMLSchema#date	M
http://www.w3.org/2001/XMLSchema#dateTime	M
urn:oasis:names:tc:xacml:2.0:data-types:dayTimeDuration	M
urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration	M
http://www.w3.org/2001/XMLSchema#anyURI	M
http://www.w3.org/2001/XMLSchema#hexBinary	M
http://www.w3.org/2001/XMLSchema#base64Binary	M
urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name	M
urn:oasis:names:tc:xacml:1.0:data-type:x500Name	M

7.8.8 函数

执行情况必须正确处理那些与标记着“M”的标识符相关联的函数。

函数	M/O
urn:oasis:names:tc:xacml:1.0:function:string-equal	M
urn:oasis:names:tc:xacml:1.0:function:boolean-equal	M
urn:oasis:names:tc:xacml:1.0:function:integer-equal	M
urn:oasis:names:tc:xacml:1.0:function:double-equal	M
urn:oasis:names:tc:xacml:1.0:function:date-equal	M
urn:oasis:names:tc:xacml:1.0:function:time-equal	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-equal	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-equal	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-equal	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal	M
urn:oasis:names:tc:xacml:1.0:function:integer-add	M
urn:oasis:names:tc:xacml:1.0:function:double-add	M
urn:oasis:names:tc:xacml:1.0:function:integer-subtract	M
urn:oasis:names:tc:xacml:1.0:function:double-subtract	M
urn:oasis:names:tc:xacml:1.0:function:integer-multiply	M
urn:oasis:names:tc:xacml:1.0:function:double-multiply	M
urn:oasis:names:tc:xacml:1.0:function:integer-divide	M
urn:oasis:names:tc:xacml:1.0:function:double-divide	M
urn:oasis:names:tc:xacml:1.0:function:integer-mod	M
urn:oasis:names:tc:xacml:1.0:function:integer-abs	M
urn:oasis:names:tc:xacml:1.0:function:double-abs	M
urn:oasis:names:tc:xacml:1.0:function:round	M
urn:oasis:names:tc:xacml:1.0:function:floor	M
urn:oasis:names:tc:xacml:1.0:function:string-normalize-space	M
urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case	M
urn:oasis:names:tc:xacml:1.0:function:double-to-integer	M
urn:oasis:names:tc:xacml:1.0:function:integer-to-double	M
urn:oasis:names:tc:xacml:1.0:function:or	M
urn:oasis:names:tc:xacml:1.0:function:and	M
urn:oasis:names:tc:xacml:1.0:function:n-of	M
urn:oasis:names:tc:xacml:1.0:function:not	M
urn:oasis:names:tc:xacml:1.0:function:integer-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal	M

函数	M/O
urn:oasis:names:tc:xacml:1.0:function:integer-less-than	M
urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:double-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:double-less-than	M
urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration	M
urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration	M
urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration	M
urn:oasis:names:tc:xacml:1.0:function:string-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:string-less-than	M
urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:time-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:time-less-than	M
urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal	M
urn:oasis:names:tc:xacml:2.0:function:time-in-range	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:date-greater-than	M
urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:date-less-than	M
urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal	M
urn:oasis:names:tc:xacml:1.0:function:string-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:string-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:string-is-in	M
urn:oasis:names:tc:xacml:1.0:function:string-bag	M
urn:oasis:names:tc:xacml:1.0:function:boolean-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:boolean-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:boolean-is-in	M
urn:oasis:names:tc:xacml:1.0:function:boolean-bag	M
urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:integer-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:integer-is-in	M
urn:oasis:names:tc:xacml:1.0:function:integer-bag	M
urn:oasis:names:tc:xacml:1.0:function:double-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:double-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:double-is-in	M
urn:oasis:names:tc:xacml:1.0:function:double-bag	M
urn:oasis:names:tc:xacml:1.0:function:time-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:time-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:time-is-in	M
urn:oasis:names:tc:xacml:1.0:function:time-bag	M
urn:oasis:names:tc:xacml:1.0:function:date-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:date-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:date-is-in	M

函数	M/O
urn:oasis:names:tc:xacml:1.0:function:date-bag	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-is-in	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-bag	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-is-in	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-bag	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-is-in	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-bag	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-is-in	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-bag	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-is-in	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-bag	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-is-in	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-bag	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-is-in	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-bag	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-is-in	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-bag	M
urn:oasis:names:tc:xacml:2.0:function:string-concatenate	M
urn:oasis:names:tc:xacml:2.0:function:uri-string-concatenate	M
urn:oasis:names:tc:xacml:1.0:function:any-of	M
urn:oasis:names:tc:xacml:1.0:function:all-of	M
urn:oasis:names:tc:xacml:1.0:function:any-of-any	M
urn:oasis:names:tc:xacml:1.0:function:all-of-any	M
urn:oasis:names:tc:xacml:1.0:function:any-of-all	M
urn:oasis:names:tc:xacml:1.0:function:all-of-all	M
urn:oasis:names:tc:xacml:1.0:function:map	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-match	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match	M
urn:oasis:names:tc:xacml:1.0:function:string-regexp-match	M
urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match	M
urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match	M
urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match	M
urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match	M
urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match	M
urn:oasis:names:tc:xacml:1.0:function:xpath-node-count	O
urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal	O

函数	M/O
urn:oasis:names:tc:xacml:1.0:function:xpath-node-match	O
urn:oasis:names:tc:xacml:1.0:function:string-intersection	M
urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:string-union	M
urn:oasis:names:tc:xacml:1.0:function:string-subset	M
urn:oasis:names:tc:xacml:1.0:function:string-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:boolean-intersection	M
urn:oasis:names:tc:xacml:1.0:function:boolean-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:boolean-union	M
urn:oasis:names:tc:xacml:1.0:function:boolean-subset	M
urn:oasis:names:tc:xacml:1.0:function:boolean-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:integer-intersection	M
urn:oasis:names:tc:xacml:1.0:function:integer-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:integer-union	M
urn:oasis:names:tc:xacml:1.0:function:integer-subset	M
urn:oasis:names:tc:xacml:1.0:function:integer-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:double-intersection	M
urn:oasis:names:tc:xacml:1.0:function:double-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:double-union	M
urn:oasis:names:tc:xacml:1.0:function:double-subset	M
urn:oasis:names:tc:xacml:1.0:function:double-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:time-intersection	M
urn:oasis:names:tc:xacml:1.0:function:time-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:time-union	M
urn:oasis:names:tc:xacml:1.0:function:time-subset	M
urn:oasis:names:tc:xacml:1.0:function:time-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:date-intersection	M
urn:oasis:names:tc:xacml:1.0:function:date-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:date-union	M
urn:oasis:names:tc:xacml:1.0:function:date-subset	M
urn:oasis:names:tc:xacml:1.0:function:date-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-intersection	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-union	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-subset	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-intersection	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-union	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-subset	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-intersection	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-union	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-subset	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-intersection	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-union	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-subset	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-intersection	M

函数	M/O
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-union	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-subset	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-intersection	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-union	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-subset	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-intersection	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-union	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-subset	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-intersection	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-union	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-subset	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-set-equals	M

8 基于核心和层次角色的接入控制(RBAC)轮廓

本部分为 XACML 的使用定义了轮廓，以满足“核心”和“层次”的基于角色接入控制（RBAC）的需求。

8.1 RBAC背景

本部分是资料性的。

本部分为 XACML 的使用定义了轮廓，以满足“核心”和“层次”的基于角色接入控制（RBAC）的需求。

注一 RBAC的信息见[RBAC]。

8.1.1 范围

基于角色的接入控制允许按照主体角色而不是严格按照单个主体身份来规定策略，这对于接入控制系统的可升级性和易管理性都很重要。

本轮廓中规定的策略可以回答三种类型的问题：

- 1) 如果一个主体具有激活的角色 R_1, R_2, \dots, R_n ，那么主体X是否能够使用一个给定的动作访问一个给定的资源？
- 2) 是否允许主体X激活角色 R_i ？
- 3) 如果一个主体具有激活的角色 R_1, R_2, \dots, R_n ，是否意味着该主体具备了与给定的角色R'相关联的许可？即，角色R'是否等于或者小于角色集合 R_1, R_2, \dots, R_n ？

本轮廓中规定的策略不回答“主体 X 具有什么角色？”，这个问题由角色授权管理来回答，不由 XACML PDP 直接处理。角色授权管理可以使用 XACML 策略，但还需要其它信息。

本轮廓中规定的策略假定一个给定主体的所有角色已经在请求授权时激活了。它们不处理角色必须基于主体试图执行的资源或动作而动态激活的环境。因此，本轮廓中规定的策略也不处理静态或者动态“职责分离”。未来的轮廓可能会解决这类环境需求。

8.1.2 角色

本建议书中角色用 XACML 主体属性来表示。有两种例外：在 Role Assignment <PolicySet>或<Policy>，和 HasPrivilegesOfRole <Policy>之中，这两种情况下角色会作为资源属性出现。

根据应用环境的需求,角色属性可以用两种方式的任意一种来表示。在某些环境中可能会有一部分“角色属性”,其中每个属性的名字是一些指示“角色”的名称,每个这样的属性的值指示了角色具有的名称。例如,在第一种环境下,可能会有一个具有 AttributeId 为"&role;"的“角色属性”(本轮廓建议使用这个标识符)。某个可能存在角色是这一属性的值,可能是"&roles;officer", "&roles;manager"和"&roles;employee"。这种表示角色的方法与 XACML 表示策略的方法一起可以很好的起作用。这种标志角色的方法也非常有益于互操作性。

另外,在其它应用环境中,可能会有许多种不同的属性标识符,每个标识符指示不同的角色。例如,在第二类环境中,可能有三个属性标识符: "urn:someapp:attributes:officer-role", "urn:someapp:attributes:manager-role", 和"urn:someapp:attributes:employee-role"。这种情况下,属性值可以为空或者包含与角色相关联的各种参数。XACML 策略可以处理用这种方法表示的角色,但是本质上与对第一种方法的处理不同。

XACML 支持单个接入请求对多个主体,以指示发出请求可能包含的各种实体。例如,通常会有个人用户发出请求,至少是间接地发出请求;通常会有一个或多个应用软件或代码库来代表用户产生实际的低层接入请求;会有执行应用软件或代码库的有着诸如 IP 地址标识的计算设备。XACML 使用标示被描述主体类型的 SubjectCategory xml 属性来标识每个 Subject。例如,个人用户具有 &subject-category;access-subject 值的 SubjectCategory (这是缺省类别) 属性;产生接入请求的应用具有 &subject-category;codebase 值的 SubjectCategory 类别等等。在本轮廓中,角色属性可以与发出接入请求的主体的任一类别相关联。

8.1.3 策略

本建议书规定四类策略:

- 1) **Role** <PolicySet> 或 **RPS**: <PolicySet>与具有Permission <PolicySet>的给定角色属性和属性值的持有者相关联, Permission <PolicySet>包含了与给定角色关联的实际许可。Role <PolicySet>的<Target>元素将<PolicySet>的适用性限制在持有相关角色属性和值的主体。每个Role <PolicySet>引用单个相应的Permission <PolicySet>,但它并不包含或引用任何其它的<Policy>或<PolicySet>元素。
- 2) **Permission** <PolicySet>或 **PPS**: <PolicySet>包含与给定角色相关联的实际的许可。<PolicySet>包含<Policy>元素和<Rules>,它们描述了主体允许访问的资源 and 动作,以及访问的其它更进一步条件,比如一天的某个时刻。给定的Permission <PolicySet>也可以包含与其它角色相关联的Permission <PolicySet>的引用信息,其中其它角色比给定角色的等级低,因此允许给定的Permission <PolicySet>继承与引用的Permission <PolicySet>的角色相关联的所有许可。如果Permission <PolicySet>的<Target>元素存在,那么<Target>元素一定不限制<PolicySet>所适用的主体。
- 3) **Role Assignment** <Policy>或<PolicySet>: <Policy>或<PolicySet>规定了角色可以指派给哪个主体,或者角色可以被哪个主体激活。它也有可能对角色的集合,或者指派给给定主体或被给定主体所激活的角色的总数进行限制。这种策略由角色激活管理来使用。使用Role Assignment <Policy> 或 <PolicySet>是可选的。
- 4) **HasPrivilegesOfRole** <Policy>: Permission <PolicySet>中的<Policy>支持询问主体是否具有权限的请求。如果要支持这类请求,每个Permission <PolicySet>都必须包括HasPrivilegesOfRole <Policy>。支持这类<Policy>,以及支持询问主体是否具有与给定角色相关联的权限的请求,是可选的。

必须把 Permission <PolicySet>实例保存在策略库中,这样它们才永远不会被当作 XACML PDP 的初始策略来使用; Permission <PolicySet>实例一定只能通过相应的 Role <PolicySet>来获得。这是因为,为了支持层次的角色, Permission <PolicySet>必须可应用于每个主体。Permission <PolicySet>依据相应的 Role <PolicySet>来保证只有持有相应角色属性的主体才可以获得给定 Permission <PolicySet>中的访问许可。

Role <PolicySet>和 Permission <PolicySet>实例的单独使用允许对层次 RBAC 的支持,这样,一个高等级角色就可以获得一个低等级角色的许可。不引用其它 Permission <PolicySet>元素的 Permission <PolicySet>实际上就是 XACML <Policy>,而不是<PolicySet>。但是,如果要求 Permission <PolicySet>

是<PolicySet>的话，就要允许它的关联角色后来成为角色层次的一部分，而不需要其它策略有任何改变。

8.1.4 多角色许可

本轮廓可能会表示一个用户同时持有几个角色而获得访问给定许可的策略。例如，改变一个医院病人的护理方法可能需要执行该动作的主体具有医师和职工双重角色。

这些策略可以用 Role <PolicySet>来表示。此时 Role <PolicySet>中的<Target>元素需要主体具有所有必需的角色属性，这可以通过使用一个包含多个<SubjectMatch>的单个<Subject>元素来实现。相关联的 Permission <PolicySet>必须规定与主体关联的许可，这些主体同时具有全部规定的激活角色。

与多角色策略关联的 Permission <PolicySet>可以引用与其它角色相关联的 Permission <PolicySet>实例，这样就可以从其它角色继承许可。如果其它角色在其自身的 Permission <PolicySet>中包含对一个与多角色策略相关联的 Permission <PolicySet>的引用，那么与给定的多角色<PolicySet>相关联的许可也可以被其它角色所继承。

8.2 RBAC例子

本部分是资料性的。

本部分给出与基于角色接入控制相关联的策略类型的完整例子。

假设一个组织使用两个角色，经理和雇员。在本例中，他们由一个 AttributeId 为"&role;"的单个 XACML 属性的两个独立值来表示。与这两个角色相对应的 &role; 属性值是 "&roles;employee" 和 "&roles;manager"。雇员可以生成购买订单。经理可以签署订单，并具有与雇员角色相关联的所有许可。因此，经理角色比雇员角色高级，雇员角色比经理角色低级。

根据本轮廓，有两个 Permission <PolicySet>实例，一个是经理角色，一个是雇员角色。经理 Permission <PolicySet>应赋予主体签署购买订单的给定许可，并引用雇员 Permission <PolicySet>以继承它的许可。雇员 Permission <PolicySet>应赋予所有主体生成购买订单的许可。

根据本轮廓，有两个 Role <PolicySet>实例：一个是经理角色，一个是雇员角色。经理 Permission <PolicySet>中包含一个<Target>，<Target>要求 Subject 的&role;属性值为"&roles;manager"。经理 Role <PolicySet>应引用经理 Permission <PolicySet>。雇员 Role <PolicySet>中包含一个<Target>，<Target>要求 Subject 的&role;属性值为"&roles;employee"。Role <PolicySet>应引用雇员 Permission <PolicySet>。

8.2.1 经理角色的Permission <PolicySet>

下列 Permission <PolicySet>包含与经理角色关联的许可。必须建立 PDP 的策略检索，以至于只有通过对经理 Role <PolicySet>的引用才能访问 Permission <PolicySet>（见表 8-1）。

表 8-1/X.1142—经理Permission <PolicySet>

```

<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
PolicySetId="PPS:manager:role"
PolicyCombiningAlgId="&policy-combine;Permit-overrides">

<!-- Permissions specifically for the manager role -->
<Policy PolicyId="Permissions:specifically:for:the:manager:role"
RuleCombiningAlgId="&rule-combine;Permit-overrides">
  <!-- Permission to sign a purchase order -->
  <Rule RuleId="Permission:to:sign:a:purchase:order" Effect="Permit">
    <Target>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="&function;string-equal">
            <AttributeValue
              DataType="&xml;string">purchase
order</AttributeValue>
            <ResourceAttributeDesignator
              AttributeId="&resource;resource-id"
              DataType="&xml;string"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="&function;string-equal">
            <AttributeValue
              DataType="&xml;string">sign</AttributeValue>
            <ActionAttributeDesignator
              AttributeId="&action;action-id"
              DataType="&xml;string"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  </Rule>
</Policy>
<!-- Include permissions associated with employee role -->
<PolicySetIdReference>PPS:employee:role</PolicySetIdReference>
</PolicySet>

```

8.2.2 雇员角色的Permission <PolicySet>

下列 Permission <PolicySet>包含与雇员角色关联的许可（见表 8-2）。必须建立 PDP 的策略检索，以至于只有引用雇员 Role <PolicySet>，或者通过经理 Permission <PolicySet>来引用更高级的经理 Role <PolicySet>才能访问 Permission <PolicySet>。

表 8-2/X.1142—雇员Permission <PolicySet>

```

<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
  PolicySetId="PPS:employee:role"
  PolicyCombiningAlgId="&policy-combine;Permit-overrides">
  <!-- Permissions specifically for the employee role -->
  <Policy PolicyId="Permissions:specifically:for:the:employee:role"
    RuleCombiningAlgId="&rule-combine;Permit-overrides">
    <!-- Permission to create a purchase order -->
    <Rule RuleId="Permission:to:create:a:purchase:order" Effect="Permit">
      <Target>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="&function;string-equal">
              <AttributeValue
                DataType="&xml;string">purchase
order</AttributeValue>
              <ResourceAttributeDesignator
                AttributeId="&resource;resource-id"
                DataType="&xml;string"/>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="&function;string-equal">
              <AttributeValue
                DataType="&xml;string">create</AttributeValue>
              <ActionAttributeDesignator
                AttributeId="&action;action-id"
                DataType="&xml;string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>
</PolicySet>

```

8.2.3 经理角色的Role<PolicySet>

根据 Role <PolicySet>的<Target>, 下列 Role <PolicySet>只适用于持有&role; (见表 8-3) Attribute 并且属性值为"&roles;manager"的主体。<PolicySetIdReference>指与经理角色相关联的 Permission <PolicySet>。

表 8-3/X.1142—经理的Role <PolicySet>

```

<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
  PolicySetId="RPS:manager:role"
  PolicyCombiningAlgId="&policy-combine;Permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="&function;anyURI-equal">
          <AttributeValue
            DataType="&xml;anyURI">&roles;manager</AttributeValue>
          <SubjectAttributeDesignator AttributeId="&role;"
            DataType="&xml;anyURI"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <!-- Use permissions associated with the manager role -->
  <PolicySetIdReference>PPS:manager:role</PolicySetIdReference>
</PolicySet>

```

8.2.4 雇员角色的Role <PolicySet>

根据 Role <PolicySet>的<Target>, 下列 Role <PolicySet>只适用于持有&role; (见表 8-4) Attribute 属性值为"&roles; employee"的主体。<PolicySetIdReference>指与雇员角色相关联的 Permission <PolicySet>。

表 8-4/X.1142—雇员的Role <PolicySet>

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
  PolicySetId="RPS:employee:role"
  PolicyCombiningAlgId="&policy-combine;Permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="&function;anyURI-equal">
          <AttributeValue
            DataType="&xml;anyURI">&roles;employee</AttributeValue>
          <SubjectAttributeDesignator AttributeId="&role;"
            DataType="&xml;anyURI"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <!-- Use permissions associated with the employee role -->
  <PolicySetIdReference>PPS:employee:role</PolicySetIdReference>
</PolicySet>
```

8.2.5 HasPrivilegesOfRole策略和请求

XACML RBAC 系统可以选择支持"该主体具有角色 X 的特权吗?"形式的查询。如果支持的话, 每个 Permission <PolicySet>必须包含 HasPrivilegesOfRole<Policy>。对于经理的 Permission<PolicySet>, 其 HasPrivilegesOfRole<Policy>如表 8-5 所示。

表 8-5/X.1142—经理的Permission <PolicySet>

```
<!-- HasPrivilegesOfRole Policy for manager role -->
<Policy PolicyId="Permission:to:have:manager:role:permissions"
  RuleCombiningAlgId="&rule-combine;Permit-overrides">
  <!-- Permission to have manager role permissions -->
  <Rule RuleId="Permission:to:have:manager:permissions" Effect="Permit">
    <Condition>
      <Apply FunctionId="&function;and">
        <Apply FunctionId="&function;anyURI-is-in">
          <AttributeValue
            DataType="&xml;anyURI">&roles;manager</AttributeValue>
          <ResourceAttributeDesignator AttributeId="&role;"
            DataType="&xml;anyURI"/>
        </Apply>
        <Apply FunctionId="&function;anyURI-is-in">
          <AttributeValue
            DataType="&xml;anyURI">&actions;hasPrivilegesofRole</AttributeValue>
          <ActionAttributeDesignator AttributeId="&action;action-id"
            DataType="&xml;anyURI"/>
        </Apply>
      </Apply>
    </Condition>
  </Rule>
</Policy>
```

对于雇员的 Permission <PolicySet>, 其 HasPrivilegesOfRole <Policy>如表 8-6 所示。

表 8-6/X.1142—雇员的Permission <PolicySet>

```

<!-- HasPrivilegesOfRole Policy for employee role -->
<Policy PolicyId="Permission:to:have:employee:role:permissions"
  RuleCombiningAlgId="&rule-combine;Permit-overrides">
<!-- Permission to have employee role permissions -->
  <Rule RuleId="Permission:to:have:employee:permissions" Effect="Permit">
    <Condition>
      <Apply FunctionId="&function;and">
        <Apply FunctionId="&function;anyURI-is-in">
          <AttributeValue
            DataType="&xml;anyURI">&roles;employee</AttributeValue>
          <ResourceAttributeDesignator AttributeId="&role;"
            DataType="&xml;anyURI"/>
        </Apply>
        <Apply FunctionId="&function;anyURI-is-in">
          <AttributeValue
            DataType="&xml;anyURI">&actions;hasPrivilegesofRole
          </AttributeValue>
          <ActionAttributeDesignator AttributeId="&action;action-id"
            DataType="&xml;anyURI"/>
        </Apply>
      </Apply>
    </Condition>
  </Rule>
</Policy>

```

询问主体 Anne 是否具有与&roles;manager 相关联的权限的请求，如表 8-7 所示。

表 8-7/X.1142—有关主体的请求

```

<Request>
  <Subject>
    <Attribute AttributeId="&subject;subject-id" DataType="&xml;string">
      <AttributeValue>Anne</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="&role;" DataType="&xml;anyURI">
      <AttributeValue>&roles;manager</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="&action;action-id"
      DataType="&xml;anyURI">&actions;hasPrivilegesOfRole</AttributeValue>
    </Attribute>
  </Action>
</Request>

```

<Request>必须包含 Anne 的直接角色（在这个例子中是&roles;employee），否则的话，PDP 的上下文处理器就必须能够发现这些直接角色。HasPrivilegesOfRole 策略不进行对角色和主体的相关联的工作。

8.3 指派和激活角色属性

本部分是资料性的。

为用户指派不同的角色属性，并在一个会话中激活这些属性不在 XACML PDP 的范围之内。必须由一个或者多个单独的实体，如角色激活管理，来完成这些功能。本轮廓假设在请求接入决定时，给定用户（主体）的角色属性出现在 XACML Request 上下文中是一种有效的指派。

那么主体的角色属性从哪里得到呢？这些角色激活管理中的某一个是什么样？该答案由实现来决定，但是可以建议一些可能性。

在一些情况下，角色属性可能来自身份管理服务。身份管理服务维护用户信息，其中包括主体被指派或许可的角色；身份管理服务充当角色激活机构。身份管理服务可能会在 LDAP 目录中存储静态角色属性，同时 PDP 的上下文处理器可以从那里检索它们。或者该服务可能会响应来自 PDP 内容处理器的有关主体角色属性的请求，请求的形式为 SAML 属性查询。

角色激活管理可以使用 XACML 角色指派<Policy>或<PolicySet>, 来决定一个主体是否被许可具有某个特别的激活的角色属性和值。Role Assignment <Policy>或 <PolicySet>回答 “主体 X 是否被许可具有已激活的角色 R_i ?” 的问题, 它不回答 “主体 X 被许可激活哪个角色?” 的问题。角色激活管理必须有一些方法可以知道哪个或哪些角色提交了请求。例如, 角色激活管理可能会维护一张所有可能角色的列表, 这样当被询问与给定主体相关联的属性时, 它可以根据角色指派策略来对每个候选角色进行请求。

本轮廓中, 角色指派策略不同于 Role <PolicySet>和 Permission <PolicySet>实例, 这些实例用于决定与每个角色相关联的接入许可。角色指派策略仅用于来自角色激活管理的 XACML Request 的情况。这种区分可用多种方法来管理, 诸如使用具有不同策略库的不同 PDP, 或者要求角色激活查询的 <Request>元素包含 SubjectCategory 为 "&subject-category;role-enablement-authority"的 <Subject>。

Role Assignment <Policy>没有固定的形式。下面的例子 (表 8-8) 只是举例了一种可能的形式。例中包含了两个 XACML <Rule>元素。第一个<Rule>声明 Anne、Seth 和 Yassir 在上午 9 点和下午 5 点之间被许可具有激活的"&roles;employee"角色。第二个<Rule>声明 Steve 具有激活的"&roles;manager"角色, 没有工作日时间的限制。

表 8-8/X.1142—Role 指派例子

```
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
  PolicyId="Role:Assignment:Policy"
  RuleCombiningAlgId="&rule-combine;Permit-overrides">
<!-- Employee role requirements rule -->
  <Rule RuleId="employee:role:requirements" Effect="Permit">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="&function;string-equal">
            <AttributeValue DataType="&xml:string">Seth</AttributeValue>
            <SubjectAttributeDesignator AttributeId="&subject;subject-id"
              DataType="&xml:string"/>
          </SubjectMatch>
        </Subject>
        <Subject>
          <SubjectMatch MatchId="&function;string-equal">
            <AttributeValue DataType="&xml:string">Anne</AttributeValue>
            <SubjectAttributeDesignator AttributeId="&subject;subject-id"
              DataType="&xml:string"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Resources>
        <Resource>
          <ResourceMatch MatchId="&function;anyURI-equal">
            <AttributeValue
              DataType="&xml:anyURI">&roles;employee</AttributeValue>
            <ResourceAttributeDesignator AttributeId="&role;"
              DataType="&xml:anyURI"/>
          </ResourceMatch>
        </Resource>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="&function;anyURI-equal">
            <AttributeValue DataType="&xml:anyURI">&actions;
              enableRole</AttributeValue>
            <ActionAttributeDesignator AttributeId="&action;action-id"
              DataType="&xml:anyURI"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
```

表 8-8/X.1142—Role 指派例子

```

<Condition>
  <Apply FunctionId="&function;and">
    <Apply FunctionId="&function;time-greater-than-or-equal">
      <Apply FunctionId="&function;time-one-and-only">
        <EnvironmentAttributeDesignator
AttributeId="&environment;current-time"
          DataType="&xml;time"/>
        </Apply>
        <AttributeValue DataType="&xml;time">9h</AttributeValue>
      </Apply>
      <Apply FunctionId="&function;time-less-than-or-equal">
        <Apply FunctionId="&function;time-one-and-only">
          <EnvironmentAttributeDesignator
AttributeId="&environment;current-time"
            DataType="&xml;time"/>
          </Apply>
          <AttributeValueDataType="&xml;time">17h</AttributeValue>
        </Apply>
      </Apply>
    </Apply>
  </Condition>
</Rule>
<!--管理者角色需求规则 le -->
<Rule RuleId="manager:role:requirements" Effect="Permit">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="&function;string-equal">
          <AttributeValue DataType="&xml;string">Steve</AttributeValue>
          <SubjectAttributeDesignator AttributeId="&subject;subject-id"
            DataType="&xml;string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="&function;anyURI-equal">
          <AttributeValue
            DataType="&xml;anyURI">&roles;:manager</AttributeValue>
          <ResourceAttributeDesignator AttributeId="&role;"
            DataType="&xml;anyURI"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="&function;anyURI-equal">
          <AttributeValue
            DataType="&xml;anyURI">&actions;enableRole</AttributeValue>
          <ActionAttributeDesignator AttributeId="&action;action-id"
            DataType="&xml;anyURI"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
</Rule>
</Policy>

```

8.4 实现RBAC的模型

本部分是资料性的。

下面各节描述了怎样使用 XACML 策略来实现 RBAC 模型（见[RBAC]）的各种组成部分。

8.4.1 核心RBAC

核心 RBAC 包括以下五个基本数据元素：

- 用户由XACML Subjects来实现。合适的时候，所有XACML SubjectCategory都可能被使用。
- 角色由一个或多个XACML主体属性来表示。一组角色是针对给定应用和给定策略域的，使用不同的角色而不产生混乱是非常重要的。为此，本轮廓虽然建议使用值为"urn:oasis:names:tc:xacml:2.0:subject:role"的通用AttributeId值，但是并不定义角色值的任何标准集。建议每个应用或者策略域达成一致发布唯一的AttributeId值，DataType值，和<AttributeValue>值集，这些值将为与其策略域对应的各种角色所使用。
- 对象用XACML Resources来表示。
- 操作用XACML Actions来表示。
- 许可用前面各节中描述的XACML Role <PolicySet> 和 Permission <PolicySet>实例来表示。

核心 RBAC 需要支持每个角色多个用户，每个用户多个角色，每个角色多个许可，和每个许可多个角色。这些需求中的每一个都可以由基于下面所述的本轮廓的 XACML 策略来满足。注意，为用户实际指派角色并不在 XACML PDP 的范围内。

XACML 允许多个主体与给定角色属性相关联。XACML Role <PolicySet>是依据主体所具有的给定角色 <Attribute> 和 <AttributeValue> 来定义的，它将应用于所有角色属性 <Attribute> 和 <AttributeValue>都在 XACML Request 上下文之中的请求用户。

XACML 允许多个角色属性或者角色属性值与给定的 Subject 属性相关联。如果 Subject 有多个激活的角色，那么应用于这些角色中任何一个角色的任何 Role <PolicySet>实例都可能被赋值，与角色相应的 Permission <PolicySet>中的许可应被通过。甚至可能会定义这样的策略，要求给定的主体同时具有多个角色属性或者激活的值。在这种情况下，与多角色需求关联的许可将应用于具有所有必需的角色属性和值的主体，同时 XACML Request 上下文被交给 PDP 来进行赋值。

与给定角色关联的 Permission <PolicySet>可能会允许使用多个动作来访问多个资源。XACML 有多种构造许可的模式，所以多权限角色可用多种方法来表示。通过在 Permission <PolicySet> B 中包括 <PolicySetIdReference>，角色 A 可以与 Permission <PolicySet> B 相关联，其中 Permission <PolicySet> B 在与角色 A 相关联的 Permission <PolicySet>中。这样，同一组许可可以与多于一个的角色相关联。

除基本核心 RBAC 需求外，使用本轮廓的 XACML 策略也可以表示与角色相关联的给定许可所应用的任意情况。这些情况可能是把许可限制在每天给定的时间段，或者限制在具有其它属性的角色持有者，不管这个是不是角色属性。

8.4.2 层次RBAC

层次 RBAC 通过定义角色间的继承关系对核心 RBAC 进行扩展。例如，可以定义角色 A 继承与角色 B 关联的所有许可。在这种情况下，角色 A 被认为在角色等级上比角色 B 低级。如果角色 B 依次继承与角色 C 关联的许可，那么由于角色 A 比角色 B 低级，角色 A 也应继承那些许可。

可以通过在 Permission <PolicySet>中包含<PolicySetIdReference>，使用本轮廓中的 XACML 策略来实现角色继承，其中 Permission <PolicySet>和与另一个角色相关联的 Permission <PolicySet>中的一个角色相关联。这样，包含<PolicySetIdReference>的角色应继承与被引用角色相关联的许可。

本配置属性采用对角色添加继承特性的方法来构造策略，在添加特性时不需要对与任何其它角色相关联的 <PolicySet>实例进行改变。组织可能不会一开始就使用角色等级，但是后来可能为了不改写已经存在的策略而决定使用这个功能。

8.5 轮廓

本部分讨论角色、角色属性、角色指派和接入控制。

8.5.1 角色和角色属性

角色用一个或多个 XACML 属性来表示。每个使用该基于角色接入控制轮廓的应用域应使用一个或者多个 AttributeId 值来定义角色属性或者对角色属性达成一致的表示方法。每个这样的 AttributeId 值与一组被许可的值以及它们的 DataTypes 相关联。为了使用策略中的相应值，AttributeId 的每个被许可的值都应具有明确的语义。

本轮廓建议所有的角色属性使用"urn:oasis:names:tc:xacml:2.0:subject:role" AttributeId 值。这个属性的实例应具有"http://www.w3.org/2001/XMLSchema#anyURI"的 DataType。

8.5.2 角色指派或激活

角色激活管理负责在一个用户会话中为用户指派和激活角色所使用的角色。它可以使用 XACML Role Assignment <Policy>或<PolicySet>来决定哪个用户在何种情况下可被许可激活哪个角色。Role Assignment <Policy>或<PolicySet>没有规定的格式。建议 Role Assignment <Policy> 或<PolicySet>中的角色用资源属性来表示，在资源属性中，AttributeId 为&role;，<AttributeValue>为相关角色值的 URI。建议指派或激活角色用动作属性来表示，在动作属性中，AttributeId 为&action;action-id，DataType 为&xml;anyURI，<AttributeValue>为&actions;enableRole。

8.5.3 接入控制

基于角色的接入控制使用两类<PolicySet>实现：Role <PolicySet>，Permission <PolicySet>。这两类<PolicySet>的特定功能和需求如下。

对于每个角色应定义一个 Role <PolicySet>。该<PolicySet>应包含<Target>元素，<Target>元素使<PolicySet>仅适用于具有与给定角色关联的 XACML 属性的主体。<Target>元素不应限制资源、动作或者环境。每个 Role <PolicySet>包含一个<PolicySetIdReference>元素，这个元素引用与角色相关联的唯一 Permission <PolicySet>。Role <PolicySet> 不应包含任何其它 <Policy>，<PolicySet>，<PolicyIdReference>，或者<PolicySetIdReference>元素。

对于每个角色应定义一个 Permission <PolicySet>。该<PolicySet>应包含<Policy>和<Rule>元素，这些元素规定具有给定角色的主体所被许可的接入类型。<PolicySet>以及它所包括或引用的<PolicySet>的<Target>，<Policy>，和<Rule>元素不应把主体限制在 Permission <PolicySet>可应用的主体上。

如果一个给定角色从一个或者多个低级的角色那里继承了许可，那么给定（高级）角色的 Permission <PolicySet>必须包含每个低级角色的<PolicySetIdReference>元素。每个这样的<PolicySetIdReference>都应引用与高级角色所继承的低级角色相关联的 Permission <PolicySet>。

Permission <PolicySet>可包含 HasPrivilegesOfRole <Policy>。这样的<Policy>应具有有“许可”作用的<Rule>元素。该规则应允许任何主体在资源上做动作，该动作具有 AttributeId 为&action;action-id，DataType 为&xml;anyUR 的属性，具有值为&actions;hasPrivilegesOfRole 的<AttributeValue>，它所在资源的属性为 Permission <PolicySet>所应用的角色（例如，AttributeId 为&role，DataType 为&xml;anyURI，<AttributeValue>值为给定角色值的 URI）。注意，角色属性，在 Role <PolicySet> <Target>中是主体属性，而在 HasPrivilegesOfRole <Policy>中被当作资源属性。

任何策略库的组织和 PDP 的配置应确保 PDP 永远不会把 Permission <PolicySet>作为 PDP 的初始策略。

8.6 标识符

本轮廓定义了以下 URN 标识符。

8.6.1 轮廓标识符

当需要一个 URI 形式的标识符时，下面的标识符应作为本轮廓的标识符被使用。

```
urn:oasis:names:tc:xacml:2.0:profiles:rbac:core-hierarchical
```

8.6.2 角色属性

下面的标识符可以当作角色属性的 AttributeId 使用。

```
urn:oasis:names:tc:xacml:2.0:subject:role
```

8.6.3 主体类别

下面的标识符可以用作主体属性的 SubjectCategory, 主体属性用以确定请求是否是来自于角色激活管理。

```
urn:oasis:names:tc:xacml:2.0:subject-category:role-enablement-authority
```

8.6.4 动作属性值

下面标识符在 HasPrivilegesOfRole <Policy> 中可以用作 &action;action-id 属性的 <AttributeValue>。

```
urn:oasis:names:tc:xacml:2.0:actions:hasPrivilegesOfRole
```

下面标识符在 Role Assignment <Policy>中 can 用作 &action;action-id 属性的 <AttributeValue>。

```
urn:oasis:names:tc:xacml:2.0:actions:enableRole
```

9 XACML的多资源轮廓

策略的赋值依据单个被请求的资源来定义，赋值由 XACML 策略决定点或者 PDP 来执行，授权决定被包含在响应上下文的单个 <Result> 元素中。然而策略执行点，或 PEP，可能会为了访问多个资源而希望提交一个请求上下文，也可能希望获得一个响应上下文，其中响应上下文为每个被请求资源包含了一个独立的授权决定 (<Result> 元素)。例如，该请求上下文可能会被用于避免在 PEP 和 PDP 之间发送多个决定请求消息。或者，PEP 可能会希望为所有层次节点提交请求上下文，也可能希望获得一个授权决定 (<Result> 元素)，授权决定指出是否主体被许可接入所有的被请求节点。例如，该请求上下文可能会用于请求者想要接入一个完整 XML 文档，或者该文档中元素的整个子树，或者有着所有子目录和文件的整个文件系统目录。

本建议书描述了 PEP 为在单一请求上下文中为多个资源而请求授权决定的三种方法，并描述了每个这样的授权决定的结果在返回给 PEP 的响应上下文中将如何表示。

本建议书也描述了 PEP 为响应层次上的所有节点的请求而请求授权决定的两种方法。

对于遵从于 XACML 的实现，对本轮廓中描述的每个机制的支持是可选的。

常用的资源属性采用以下的缩写：

- "resource-id" 属性: AttributeId 为
"urn:oasis:names:tc:xacml:1.0:resource:resource-id" 的资源属性。
- "scope" 属性: AttributeId 为
"urn:oasis:names:tc:xacml:2.0:resource:scope" 的资源属性。

该属性的更多信息见 9.3 节。

9.1 请求多个资源

本部分是标准化的，但是可选。

单一 XACML Request 上下文可以对每一个资源用独立的授权决定来表示一个访问多资源的请求。本部分规定了这种请求和响应的语法和语义。

对访问多个资源的请求进行赋值而产生的<Result>元素应与那些由一组请求，其中每个请求只要求访问一个资源，而产生的<Result>元素相同。每个这样的资源被称作独立资源。与每个<Result>元素相对应的概念性请求上下文被称作独立资源请求。<Result>元素中的 ResourceId 值在相应的独立资源请求中的"resource-id"属性应是<AttributeValue>。包含多个授权决定请求的源请求上下文与独立资源请求之间的映射，以及多个授权决定与单个响应上下文中的多个<Result>元素之间的相应映射，可以由本建议书中的上下文处理器来完成。本轮廓不要求按照前述的模式或者遵从构造的实际资源请求来赋值访问多个资源的请求。本轮廓仅要求<Result>元素是相同的，就好像前述模式被使用一样。

下面小节规定请求访问多个资源的三种方法。规定请求的每种方法描述了在响应上下文中与<Result>元素相对应的独立资源请求。

由 PEP 提交的单个 XACML Request 上下文可以使用多于一种的方法来请求访问不同<Resource>元素中的多个资源。

9.1.1 由"范围"标识的节点

本部分是标准化的，但是可选。

为规定对层次中的多个资源访问的请求，本部分描述"范围"资源属性两个值的用法。不管是不是 XML 文档，该语法都可以用于任何层次资源。

9.1.1.1 轮廓URI

下面 URI 值用作本轮廓的各个章节中规定功能的 URI 标识符。支持 XML 资源功能时，会用到第一个标识符，支持非 XML 文档的资源时，会用到第二个标识符。

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:scope:xml  
urn:oasis:names:tc:xacml:2.0:profile:multiple:scope:non-xml
```

9.1.1.2 源请求上下文语法

源 XACML Request 上下文<Resource>元素应包含值为"Children"或者为"Descendants"的"Scope"属性。如果被请求资源是 XML 文档，那么<ResourceContent>元素应出现，并包含被请求元素为其一部分的完整 XML 文档。如果被请求资源是 XML 文档，那么用作"resource-id"属性值的 XPath 表达式应对只包含一个节点的节点集进行赋值。

9.1.1.3 语义

该请求上下文应被解释为一个接入一组层次节点的请求，这一组节点与"resource-id"属性中规定的单个节点相关联。如果"范围"属性的值是"Children"，那么每个独立资源就是由"resource-id"属性（或者某些属性，其中单个资源具有多个标准化的标识符）所表示的一个节点和其所有子节点。如果"范围"属性的值是"Descendants"，那么独立资源就是由"resource-id"属性表示的一个节点和其所有子节点。

每个独立资源请求应与源请求上下文相同，除了两个例外："范围"属性不出现，和<Resource>元素表示单个独立资源。该<Resource>元素应包含至少一个"resource-id"属性，该属性的所有值都是独立资源的唯一的、标准化的标识符。如果源请求上下文中的"resource-id"属性包含一个发行者，那么独立资源请求中的"resource-id"属性应包含同样的发行者。如果<ResourceContent>元素在源请求上下文中出现，那么每个独立资源请求中应包含同样的<ResourceContent>元素。

XACML 和本轮廓都没有规定上下文处理器如何获得确定哪个节点是给定节点的子节点和子节点的必需信息，除非是 XML 文档，此时这些信息应从<ResourceContent>元素中获得。

9.1.2 由XPath标识的节点

本部分是标准化的，但是可选。

为在 XML 文档中规定接入多个节点请求，本部分描述"resource-id"属性中的 XPath 表达式的用法，以及"范围"属性中的"XPath-expression"值的用法。该语法用于 XML 文档资源。

9.1.2.1 轮廓URI

下面 URI 值用于在本轮廓的部分中规定功能的 URI 标识符：

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:xpath-expression
```

9.1.2.2 源请求上下文

源 XACML Request 上下文 <Resource> 元素应包含 <ResourceContent> 元素和 DataType 为 "urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression" 的 "resource-id" 属性，这样 "resource-id" 属性的 <AttributeValue> 就是一个对节点集进行赋值的 XPath 表达式，其中节点集表示 <ResourceContent> 中的多个节点。<Resource> 元素应包含具有 "XPath-expression" 值的 "范围" 属性。

9.1.2.3 语义

该请求上下文应被解释为一个接入节点集中多个节点请求，其中节点集由 "resource-id" 属性的 <AttributeValue> 来表示。每个这样的节点表示一个独立资源。

每个独立资源请求应与源请求上下文相同，除非两个例外："范围"属性不出现，和 "resource-id" 属性值是赋值 <ResourceContent> 元素中的单个节点的 XPath 表达式。该单个节点应是独立资源。如果源请求上下文中的 "resource-id" 属性包含一个发行者，那么独立资源请求中的 "resource-id" 属性应包含相同的发行者。

9.1.3 多个<Resource>元素

本部分是标准化的，但是可选。

本部分描述请求上下文中的多个 <Resource> 元素的用法，以规定访问多个资源的请求。不管是不是 XML 文档，是不是层次的资源，该语法都可以用于任何资源。

9.1.3.1 轮廓URI

下面 URI 值用于在本轮廓的部分中规定功能的 URI 标识符：

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:multiple-resource-elements
```

9.1.3.2 源请求上下文

XACML Request 上下文包含多个 <Resource> 元素。

9.1.3.3 语义

该请求上下文应被解释为访问个体 <Resource> 元素中规定的所有资源的一个请求。每个 <Resource> 元素应代表一个独立资源，除非元素使用本轮廓描述的其它机制。

应为每个 <Resource> 元素创建一个独立资源请求。该独立资源请求应与源请求上下文相同，除非一个例外：<Resource> 元素出现。如果该 <Resource> 元素包含具有除 "Immediate" 之外任何值的 "范围" 属性，那么独立资源请求应根据本轮廓的相应部分被进一步处理。该处理可以在 PDP 作出赋值前将一个独立资源请求分解到其它独立资源请求中。

9.2 请求整个层次

本部分是标准化的，但是可选。

在一些情况下，资源是层次化的，但是授权决定请求会想要请求访问资源中的所有节点，或者访问资源中的节点的整个子层次。这种情况可能是为了拷贝整个文档而请求访问 XML 文档，或者在请求所有子目录和文件时访问整个文件系统目录。需要单个<Result>来指示请求者是否被许可接入整个节点集。

由赋值这样一个接入请求而生成的<Result>元素应与通过下面处理而生成的<Result>元素相同。一组请求上下文被赋值，其中每个请求只请求接入层次节点中的一个。当且仅当对个体节点赋值而得到的所有<Result>元素中包含了"Permit"的<Decision>，返回给 PEP 的<Result>中的<Decision>才可以是"Permit"。否则，返回给 PEP 的<Result>中的<Decision>应是"Deny"。本轮廓不要求按照前述的模式或者遵循构造的实际资源请求来赋值接入这样一个层次资源的请求。本轮廓仅要求<Result>元素应是相同的，就好像前述模式被使用一样。

下面小节规定这种功能的两种语法，一个是用于 XML 文档资源，另一个是用于非 XML 文档资源。

9.2.1 XML资源

本部分是标准化的，但是可选。

本部分描述使用 XML 文档中的所有递归子元素来请求访问整个 XML 文档或者文档中的一个元素的语法。

9.2.1.1 轮廓URI

下面 URI 值用于在本轮廓的部分中规定功能的 URI 标识符：

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:entire-hierarchy.xml
```

9.2.1.2 源请求上下文

源请求上下文中的<Resource>元素应包含值为"EntireHierarchy"的"scope"属性。

源请求上下文中的<Resource>元素应包含 DataType 为"urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression"的"resource-id"属性，这样赋值节点集的<AttributeValue>只表示<ResourceContent>元素中的一个节点。

源请求上下文中的<Resource>元素可以包含其它属性。

9.2.1.3 语义

该请求的<Result>应与下面处理所生成的<Result>相同。上下文处理器应为被请求的层次上的每个节点创建一个新的请求上下文。每个这样的请求上下文应包含一个具有 DataType 为"urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression"的"resource-id"属性的单个<Resource>，和一个赋值节点集的 XPath 表达式的值，该节点集只包含<ResourceContent>元素中的一个节点。上下文处理器应向 PDP 提交每个这样的新请求上下文用以赋值，并应与相应<Result>元素中的<Decision>保持一致。当且仅当所有新的请求上下文赋值为"Permit"，那么包含<Decision>为"Permit"的单个<Result>才会被放进返回给 PEP 的响应上下文。如果新请求上下文中的任何一个赋值为"Deny"、"Indeterminate"、或"NotApplicable"，那么包含<Decision>为"Deny"的单个<Result>应被放进返回给 PEP 的响应上下文中。

9.2.2 非XML资源

本部分是标准化的，但是可选。

本部分描述请求接入非 XML 文档的层次资源中的整个节点层次的语法。

9.2.2.1 轮廓URI

下面 URI 值用于在本轮廓的部分中规定功能的 URI 标识符:

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:entire-hierarchy:non-xml
```

9.2.2.2 源请求上下文

源请求上下文中的<Resource>元素应包含值为"EntireHierarchy"的"scope"属性。

源请求上下文中的<Resource>元素应包含代表层次资源中单个节点的单个"resource-id"属性。

源请求上下文中的<Resource>元素可能包含其它属性。

由 XACML 轮廓为本建议书中的层次资源所规定的层次资源中的节点的表示法可以用于表示单个节点的身份。

9.2.2.3 语义

该请求的<Result>应与下面处理所生成的<Result>相同。内容处理器会为被请求层次中的每个节点创建一个新的请求上下文。每个这样的请求上下文应包含单个<Resource>元素，<Resource>元素具有一个值为层次中节点的身份的"resource-id"属性。上下文处理器向 PDP 提交每个这样的新请求上下文用以赋值，并与相应<Result>元素中的<Decision>保持一致。当且仅当所有新的请求上下文赋值为"Permit"，包含<Decision>为"Permit"的单个<Result>才被放进返回给 PEP 的响应上下文。如果新请求上下文中的任何一个赋值为"Deny"、"Indeterminate"或"NotApplicable"，那么包含<Decision>为"Deny"的单个<Result>应被放进返回给 PEP 的响应上下文中。

XACML 和本轮廓都没有说明上下文处理器如何获得那些确定哪个节点是最初规定节点的子节点所必需的信息，也没有规定如何表示每个这样节点的身份。本建议书中 XACML 轮廓规定的层次资源中的节点表示法可以用于表示每个这样节点的身份。

9.3 新属性标识符

下面标识符被用作资源属性的 AttributeId，资源属性指明了请求上下文的单个<Resource>元素中用于接入请求的范围("scope"属性)。

```
urn:oasis:names:tc:xacml:2.0:resource:scope
```

该属性的 DataType 应为"http://www.w3.org/2001/XMLSchema#string"。

该属性的有效值在下面和 7.5 节中列出。一次实现可以支持这些值的任何子集，包括空集。

- "Immediate" — <Resource>元素指单个非层次资源或者层次资源中的单个节点。如果"范围"属性不出现的话，该值为缺省值。<Resource>元素应按照第7节来处理。
- "Children" — <Resource>元素指层次中的多个资源。一组资源由"resource-id"资源属性所描述的单个节点和层次中的所有该节点的子节点构成。<Resource>元素按照本轮廓的9.1.1节来处理。
- "Descendants" — <Resource>元素指层次中的多个资源。一组资源由"resource-id"资源属性所描述的单个节点和层次中的所有该节点的子节点构成。<Resource>元素按照本轮廓的9.1.1节来处理。
- "XPath-expression" — <Resource>元素指多个资源。一组资源由"resource-id"资源属性所描述的节点集中的节点构成。每个节点都被包含在<Resource>元素的<ResourceContent>元素中。<Resource>元素按照本轮廓的9.1.2节来处理。

- "EntireHierarchy" — <Resource>元素指单个资源。资源由"resource-id"资源属性所描述的节点以及该节点的所有子节点构成。所有节点应是包含在<Resource>元素的<ResourceContent>中的XML文档的节点。<Resource>元素按照本轮廓的9.2节来处理。

9.4 新轮廓标识符

下面 URI 值用作本轮廓的各个章节中规定功能的 URI 标识符：

- <Resource>中"children" 或 "descendants"的"scope"属性：XML资源

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:scope:xml
```

- <Resource>中"children" 或 "descendants"的"scope"属性：Non-XML资源

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:scope:non-xml
```

- "resource-id"属性中的XPath表达式

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:xpath-expression
```

- 多<Resource>元素

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:multiple-resource-elements
```

- 请求整个层次：XML资源

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:entire-hierarchy:xml
```

- 请求整个层次：Non-XML资源

```
urn:oasis:names:tc:xacml:2.0:profile:multiple:entire-hierarchy:non-xml
```

10 XACML的SAML 2.0轮廓

本节定义了一个如何使用 SAML2.0（参见 ITU-T X.1141 建议书）来保护，传输以及请求 XACML 模式定义实例以及其他 XACML 实现所需要的信息的轮廓。

SAML 是一个基于 XML 的体系框架，用来交换安全信息。安全信息用有关主体断言的形式来表示，这里主体是在某个安全域中具有身份的实体（可以是人或计算机）。单个断言可以包含关于认证、授权以及属性的几个不同的内部声明。SAML 定义了一个协议，通过该协议客户端能够从 SAML 管理那里请求断言并获得响应。这个由基于 XML 的请求和响应消息格式组成的协议能够被绑定到许多不同的底层通信传输协议中；SAML 目前定义了一种基于 HTTP 架构的 SOAP 协议上的绑定。在创建响应的过程中，SAML 管理可以使用各种来源的信息，例如外部策略材料以及他们接收到的作为输入的请求中的断言。SAML 定义了断言元素、主体、条件、建议及声明。

本节使用六类查询和声明：

- 1) AttributeQuery: 一个标准的SAML请求，用来从属性管理那里请求一个或多个属性。
- 2) AttributeStatement: 一个标准的SAML声明，包含一个或多个属性。该声明可以作为属性库中存储属性的一种形式，在来自属性管理的SAML响应中，或者SAML声明中使用。
- 3) XACMLPolicyQuery: 一个SAML请求扩展，在本轮廓中定义。它用来从一个策略管理点那里请求一个或多个策略。
- 4) XACMLPolicyStatement: 一个SAML声明扩展，在本轮廓中定义。它可以作为策略库中存储策略的一种形式，在来自策略管理点的SAML响应中或者在SAML声明中使用。
- 5) XACMLAuthzDecisionQuery: 一个SAML请求扩展，在本轮廓中定义。PEP使用它来从一个XACML PDP那里请求一个授权决定。

6) XACMLAuthzDecisionStatement: 一个SAML声明扩展, 在本轮廓中定义。它可以在来自 XACML PDP的SAML响应中使用, 它也可以在SAML声明中作为证书使用, 但这不是当前定义的 XACML使用模式的一部分。

下图(图 10-1)示出了 XACML 使用模式以及在各种组成部分之间通信所使用的消息。不是所有的组成部分都会在每个实现中用到。

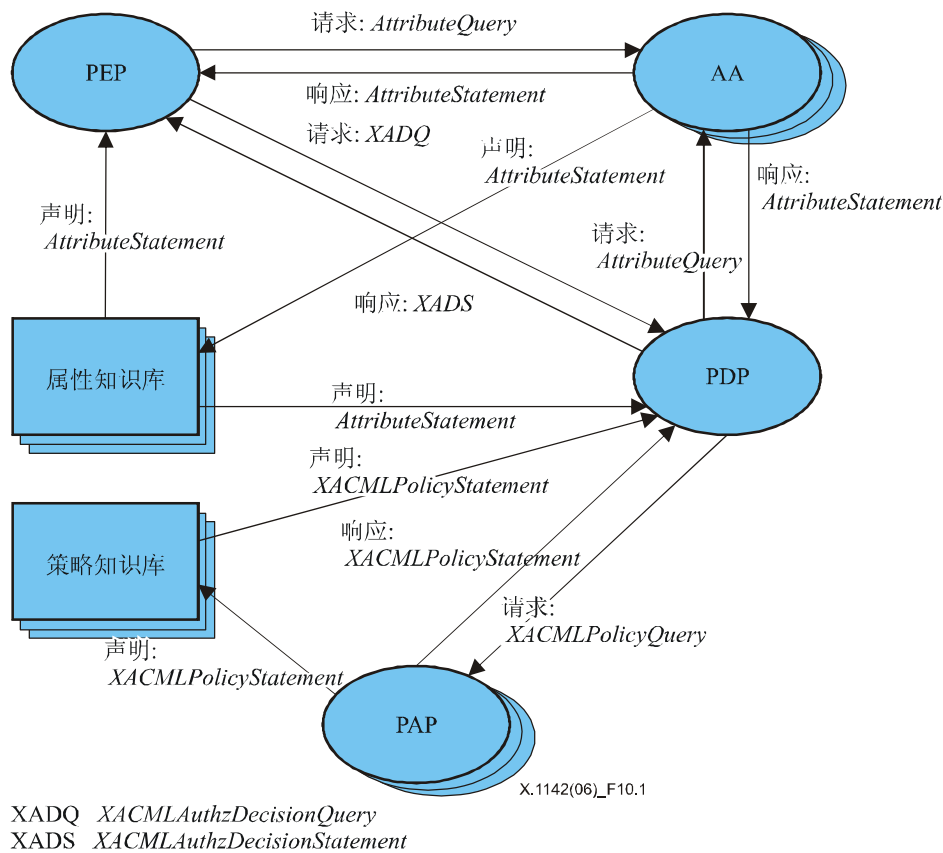


图 10-1/ X.1142—XACML使用模式

本节描述所有这些查询和声明模式定义元素, 并描述如何使用它们。同时也描述了用 XACML 来使用 SAML 的一些其他方面。本建议书不需要对 XACML 进行改变或扩展, 但确实定义了对 SAML 的扩展。

为增强可读性, 本轮廓的示例采用以下 XML 内部实体声明:

```

^lt;!ENTITY saml "urn:oasis:names:tc:SAML:2.0:assertion"
^lt;!ENTITY samlp "urn:oasis:names:tc:SAML:2.0:protocol"
^lt;!ENTITY xacml "urn:oasis:names:tc:xacml:2.0:"
^lt;!ENTITY xacml-context "urn:oasis:names:tc:xacml:2.0:context:schema:os"
^lt;!ENTITY xml "http://www.w3.org/2001/XMLSchema#"
^lt;!ENTITY subject-id "urn:oasis:names:tc:xacml:1.0:subject:subject-id"
^lt;!ENTITY resource "urn:oasis:names:tc:xacml:1.0:resource:"
^lt;!ENTITY resource-id "urn:oasis:names:tc:xacml:1.0:resource:resource-id"
^lt;!ENTITY action-id "urn:oasis:names:tc:xacml:1.0:action:action-id"
^lt;!ENTITY environment "urn:oasis:names:tc:xacml:1.0:environment:"
^lt;!ENTITY current-dateTime
    "urn:oasis:names:tc:xacml:1.0:environment:current-dateTime"

```

例如, "&xml;#string"等同于 <http://www.w3.org/2001/XMLSchema#string>。与扩展了 SAML 声明模式定义的 XACML 模式定义相关联的命名空间是:

```
xacml-saml="urn:oasis:names:tc:xacml:2.0:saml:assertion:schema:os"
```


与扩展了 SAML 协议模式定义的 XACML 模式定义相关联的命名空间是:

```
xacml-samlp="urn:oasis:names:tc:xacml:2.0:saml:protocol:schema:os"
```

10.1 映射SAML和XACML属性

SAML 声明模式定义定义了一个属性声明。SAML 协议模式定义定义了一个 AttributeQuery, 用来请求属性声明的实例并定义包含被请求实例的响应。使用 XACML 的系统可以使用这些传输及存储 SAML 属性的 SAML 元素实例。使用 XACML 的系统可以使用 SAML AttributeQuery 协议来请求 SAML 属性实例。为了能在 XACML Request 上下文中使用, SAML 属性要被映射成 XACML 属性。

SAML 属性声明是一个 <saml:Assertion> 实例, 它包含一或多个 <saml:AttributeStatement> 实例, 而每个 <saml:AttributeStatement> 实例又可能包含一或多个 <saml:Attribute> 实例。

为了能在 XACML Request 上下文中使用, SAML 属性声明中的每个 SAML 属性都要遵守 ITU-T X.1141 建议书 中的 XACML 属性轮廓, 其命名空间为 urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML。

<xacml-context:Attribute> 应从相应的 SAML 属性声明中的 <saml:Attribute> 元素中构建, 如下所示。

- XACML AttributeId XML 属性
 - 应使用完全合格的 <saml:Attribute> Name XML 属性值。
- XACML DataType XML 属性
 - 应使用完全合格的 <saml:Attribute> DataType XML 属性值。如果缺少 <saml:Attribute> DataType XML 属性, 则 XACML DataType XML 属性是 <http://www.w3.org/2001/XMLSchema#string>。
- XACML Issuer XML 属性
 - 应使用 SAML Attribute Assertion 中 <saml:Issuer> 元素的字符串值。
- <xacml-context:AttributeValue>
 - <saml:AttributeValue> 值应用来作为 <xacml-context:AttributeValue> 的元素值。

每个 <saml:Attribute> 实例都映射到一个单独的 <xacml-context:Attribute> 元素。不是 SAML 属性声明中的所有 <saml:Attribute> 实例都要被映射; 需要被映射的 SAML 属性实例可以通过某种机制选择, 在此不详细规定。<saml:Assertion> 元素的 Issuer 被用来作为每个 <xacml-context:Attribute> 元素为之创建的 Issuer。

从 <saml:Assertion> 中创建的 <xacml-context:Attribute> 应当置于 <xacml-context:Resource>, <xacml-context:Subject>, <xacml-context:Action> 之中, 或置于与 SAML 属性声明中的 <saml:Subject> 实体相应的 <xacml-context:Environment> 元素中。举例来说, 如果 SAML 属性声明主体包含一个 <saml:NameIdentifier> 元素, 并且该 NameIdentifier 值与 <xacml-context:Attribute> 的值相匹配, 而该 <xacml-context:Attribute> 具有一个 &resource;resource-id 的 AttributeId, 则从那个 SAML 属性声明中的 <saml:Attribute> 实例创建出的 <xacml-context:Attribute> 实例应该被放在 <xacml-context:Resource> 元素中。如果 <xacml-context:Attribute> 被放在 <xacml-context:Subject> 元素中, 那么 XACML SubjectCategory XML 属性也应和作为 <saml:Subject> 主体的实体相一致。

执行映射的实体要确保已经遵守 SAML 为 <saml:Assertion> 中的元素定义的语义。映射实体自身不需要进行语义检查, 但要确保在从 <saml:Assertion> 创建任何 <xacml:Attribute> 之前, 已经由 XACML PDP 进行了检查。这些语义检查包括但不限于以下内容。

- 对于使用了 SAML-derived <xacml:Attribute> 的 <xacml:Request> 来说, <saml:Assertion> 中的任何 NotBefore 和 NotOnOrAfter XML 属性都应是有效的。这意味着 NotBefore 和 NotOnOrAfter XML 属性值应与和 <xacml:Request> 相关联的 &environment;current-time, &environment;current-date, 以及 &environment;current-dateTime <xacml:Attribute> 值一致。

- 执行映射的实体要确保已经遵守SAML为任何<saml:AudienceRestrictionCondition>或<saml:DoNotCacheCondition>元素定义的语义。
- 如果<saml:Assertion>中出现<ds:Signature>元素，那么执行映射的实体应当确保签名有效，且SAML<Issuer>元素与签名中的任何<ds:X509IssuerName>值相一致。应当遵守ITU-T X.1141建议书中关于数字签名的指导方针。

10.2 授权决定

SAML 2.0 定义了一个基本的 AuthzDecisionQuery（参见 ITU-T X.1141 建议书）。SAML AuthzDecisionQuery 不能传送 XACML PDP 可以接受下来作为其请求环境一部分的所有信息。同样的，SAML AuthzDecisionStatement 也不能传送 XACML 响应环境中包含的所有信息。

为允许 PEP 使用完全支持 XACML 请求上下文和应答上下文语法的 SAML 请求和应答语法，本建议书定义了两个 SAML 扩展：

- <xacml-sampl:XACMLAuthzDecisionQuery> 是一个SAML查询，它扩展了SAML协议模式定义（参见ITU-T X.1141建议书）。它允许PEP在SAML请求中与其他信息一起，提交一个XACML请求上下文。
- <xacml-saml:XACMLAuthzDecisionStatement> 是一个SAML声明，它扩展了SAML声明模式定义（参见ITU-T X.1141建议书）。它允许XACML PDP在应答中与其他信息一起，返回一个XACML应答上下文给<XACMLAuthzDecisionStatement>。它也允许XACML应答上下文以SAML声明的形式被存储或传输。

10.2.1 元素<XACMLAuthzDecisionQuery>

<XACMLAuthzDecisionQuery>元素可以被 PEP 用来从 XACML PDP 那里请求一个授权决议。它允许 SAML 请求传送 XACML 请求上下文实例。

```
<xs:element name="XACMLAuthzDecisionQuery" type="XACMLAuthzDecisionQueryType"/>
<xs:complexType name="XACMLAuthzDecisionQueryType">
  <xs:complexContent>
    <xs:extension base="sampl:RequestAbstractType">
      <xs:sequence>
        <xs:element ref="xacml-context:Request"/>
      </xs:sequence>
      <xs:attribute name="InputContextOnly" type="boolean" use="optional" default="false"/>
      <xs:attribute name="ReturnContext" type="boolean" use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

<XACMLAuthzDecisionQuery>元素具有 **XACMLAuthzDecisionQueryType** 复杂类型。该元素可作为 SAML 定义的<sampl:AuthzDecisionQuery>的替代，允许 PEP 使用 XACML PDP 的所有能力。

<XACMLAuthzDecisionQuery> 元素包含以下 XML 属性和元素：

- InputContextOnly [缺省为 "false"]

该XML属性管理允许PDP用来做出授权决议的信息资源。如果该XML属性为"true"，则授权决议应该完全以<XACMLAuthzDecisionQuery>中包含的信息为基础而做出；没有外部属性可供使用。如果该XML属性为"false"，那么授权决议可以根据不包含在<XACMLAuthzDecisionQuery>中的外部属性来做出。

- ReturnContext [缺省为 "false"]

该XML属性允许PEP要求在由请求产生的<XACMLAuthzDecisionStatement>中包含一个<xacml-context:Request>元素。它同时也管理该<xacml-context:Request>元素的内容。

如果该 XML 属性为 "true"，则 PDP 应该在 <XACMLResponse> 中的 <XACMLAuthzDecisionStatement> 元素里包含 <xacml-context:Request> 元素。该 <xacml-context:Request> 元素应包含所有 PEP 在 <XACMLAuthzDecisionQuery> 中提供的，用来做出授权决议的属性。PDP 可以在该 <xacml-context:Request> 元素中包含其他属性，如 PDP 获得的用来做出授权决议的外部属性，或 PDP 了解的，可能对 PEP 做出后续 <XACMLAuthzDecisionQuery> 请求有帮助的其他属性。

如果该 XML 属性为 "false"，那么 PDP 不应在 <XACMLResponse> 的 <XACMLAuthzDecisionStatement> 元素中包含 <xacml-context:Request> 元素。

- <xacml-context:Request> [必需的]
一个 XACML 请求上下文。

10.2.2 元素 <XACMLAuthzDecisionStatement>

<XACMLAuthzDecisionStatement> 可以被 XACML PDP 用来给 PEP 返回一个包含 XACML 应答上下文的 SAML 响应，以回应 <XACMLAuthzDecisionQuery>。它也可用在 SAML 声明中，作为库中存储授权决议的一种形式。

```
<xs:element name="XACMLAuthzDecisionStatement" type="xacml-saml:XACMLAuthzDecisionStatementType"/>
<xs:complexType name="XACMLAuthzDecisionStatementType">
  <xs:complexContent>
    <xs:extension base="saml:StatementAbstractType">
      <xs:sequence>
        <xs:element ref="xacml-context:Response"/>
        <xs:element ref="xacml-context:Request" MinOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

<XACMLAuthzDecisionStatement> 元素具有 **XACMLAuthzDecisionStatementType** 复杂类型。该元素可作为 SAML 定义的 <samlp:AuthzDecisionStatement> 的替代，允许 SAML 声明包含来自 XACML PDP 的响应的所有内容。

<XACMLAuthzDecisionStatement> 元素包含以下元素：

- <xacml-context:Response> [必需的]
XACML PDP 创建的 XACML 应答上下文，以回应 <XACMLAuthzDecisionQuery>。
- <xacml-context:Request> [可选的]
<xacml-context:Request> 包含 XACML 属性，它由 XACML PDP 返回，以应答 <XACMLAuthzDecisionQuery>。如果 <XACMLAuthzDecisionQuery> 中的 ReturnResponse XML 属性为 "true"，那么应该包含该元素。如果 <XACMLAuthzDecisionQuery> 中的 ReturnResponse XML 属性为 "false"，则不应包含该元素。

10.3 策略

XACML 定义了两个策略模式定义元素：<Policy> 和 <PolicySet>。SAML 没有为策略定义任何协议或断言模式定义。本部分为 <XACMLPolicyQuery> 和 <XACMLPolicyStatement> 元素定义了新的 SAML 扩展。这些新元素的实例能被用来请求、传输和存储 XACML <Policy> 和 <PolicySet> 实例。

10.3.1 元素 <XACMLPolicyQuery>

<XACMLPolicyQuery> 元素被 PDP 用来请求一个或多个 XACML 策略，或从一个在线策略管理点处请求 PolicySet 实例作为 SAML 请求的一部分。

```
<xs:element name="XACMLPolicyQuery" type="XACMLPolicyQueryType"/>
<xs:complexType name="XACMLPolicyQueryType">
  <complexContent>
    <xs:extension base="samlp:RequestAbstractType">
```

```

        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element ref="xacml-context:Request"/>
            <xs:element ref="xacml:Target"/>
            <xs:element ref="xacml:PolicySetIdReference"/>
            <xs:element ref="xacml:PolicyIdReference"/>
        </xs:choice>
    </xs:extension>
</xs:complexContent>
</xs:complexType>

```

<XACMLPolicyQuery>元素具有 **XACMLPolicyQueryType** 复杂类型。

<XACMLPolicyQuery>元素包含一或多个下列元素：

- <xacml-context:Request> [任意数量]
提供一个 XACML 请求上下文。所有适用于该请求的 XACML 策略以及 PolicySet 实例都应被返回。
- <xacml:Target> [任意数量]
提供一个 XACML <Target> 元素。所有适用于该<Target>的 XACML 策略以及 PolicySet 实例都应被返回。
- <xacml:PolicySetIdReference> [任意数量]
标识一个要被返回的 XACML <PolicySet>。
- <xacml:PolicyIdReference> [任意数量]
标识一个要被返回的 XACML <Policy>。

10.3.2 元素<XACMLPolicyStatement>

<XACMLPolicyStatement> 被策略管理点使用，用来在 SAML 响应中返回一个或多个 XACML <Policy> 或 <PolicySet> 实例给 <XACMLPolicyQuery> SAML 请求。<XACMLPolicyStatement> 也可用在 SAML 声明中，作为在库中存储 <XACMLPolicyStatement> 的一种形式。

```

<xs:element name="XACMLPolicyStatement" type="xacml-
saml:XACMLPolicyStatementType"/>
<xs:complexType name="XACMLPolicyStatementType">
    <xs:complexContent>
        <xs:extension base="saml:StatementAbstractType">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element ref="xacml:Policy"/>
                <xs:element ref="xacml:PolicySet"/>
            </xs:choice>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

```

<XACMLPolicyStatement> 元素具有 **XACMLPolicyStatementType** 复杂类型。

<XACMLPolicyStatement> 元素包含以下元素。如果 <XACMLPolicyStatement> 是为响应 <XACMLPolicyQuery> 而产生的，并且没有 <xacml:Policy> 或 <xacml:PolicySet> 实例可以满足相关联的 <XACMLPolicyQuery> 的规范要求，那么在 <XACMLPolicyStatement> 中就不应有元素。

- <xacml:Policy> [任意数量]
满足相关联的 <XACMLPolicyQuery> 规范要求的 一个 <xacml:Policy> 实例，如果有的话。
- <xacml:PolicySet> [任意数量]
满足相关联的 <XACMLPolicyQuery> 规范要求的 一个 <xacml:PolicySet> 实例，如果有的话。

10.4 元素<saml:Assertion>

<XACMLAuthzDecisionStatement> , <XACMLPolicyStatement> , 或 SAML 标准的 <saml:AttributeStatement> 应被封装在一个可能已签名的 <saml:Assertion>中。

<saml:Assertion> 的大部分成员在 ITU-T X.1141 建议书中都已详细说明。下列元素和 XML 属性在此进一步加以说明, 以便与本轮廓中定义并使用的 SAML 声明类型一起使用。

除在此说明之外, 本轮廓不对<saml:Assertion>元素中的信息加以要求或限制。

10.4.1 元素<saml:Issuer>

<saml:Issuer> 元素是一个必需的元素, 用来表示关于“在断言中提出声明的 SAML 管理”的信息。

为支持第三方数字签名, 本轮廓不要求<saml:Issuer>元素中提供的身份与签名者的身份一致。是否与给<saml:Assertion>签名的管理具有适当的信任关系取决于信任方。

当<saml:AttributeAssertion>被用来构建 XACML 属性时, <saml:Issuer> 元素的字符串值会被用来作为 XACML Issuer XML 属性值, 在规定 SAML 值时要牢记这一点。

10.4.2 元素<ds:Signature>

<ds:Signature> 元素是一个可选的元素, 用来表示“一个认证断言的 XML 签名”。

<ds:Signature> 元素可以被用在和 XACML 声明一起使用的声明中。为了支持第三方签名, 本轮廓不要求<saml:Issuer>元素中提供的身份与签名者的身份一致。是否与给<saml:Assertion>签名的管理具有适当的信任关系取决于信任方。

信任方应当核查包含在声明中的任何签名并且不应该使用从断言中得到的信息, 除非该签名已核查成功。

10.4.3 元素<saml:Subject>

<saml:Subject> 元素是一个可选元素, 用来表示“断言中声明的主体”。

<saml:Subject> 元素不应被包括在含有<XACMLAuthzDecision>或<XACMLPolicy>的断言中。

在一个要被映射到 XACML 属性的<saml:AttributeAssertion> 中, 属性及其值应被绑定到一个实体, <saml:Subject> 元素应该包含该实体的身份。对于一个 XACML <Subject> 属性来说, 该身份应当与在同一个<Subject>元素中出现的任何 XACML &subject-id; 属性值一致。

对于一个 XACML <Resource> 属性来说, 该身份应当与在同一个<Resource>元素中出现的任何 XACML &resource-id; 属性值一致。对于一个 XACML <Action> 属性来说, 该身份应当与在同一个<Action>元素中出现的任何 XACML &action-id; 属性值一致。对于一个 XACML <Environment> 属性来说, 该身份应当与在同一个<Environment>元素中出现并提供了环境身份的任何 XACML 属性值一致。

10.4.4 元素<saml:Conditions>

<saml:Conditions> 元素是一个可选元素, 用来表示“在赋值断言有效性以及/或者使用声明过程中必须考虑的条件”。

<saml:Conditions> 元素应当包含 NotBefore 和 NotOnOrAfter XML 属性, 来规定声明有效性上的限制。如果出现这些 XML 属性, 信任方应当确保仅当请求上下文¤t-dateTime;资源属性值被包含在断言的规定有效周期时, 得自声明的信息才能被 PDP 用来赋值策略。

10.5 元素<samlp:RequestAbstractType>

<XACMLAuthzDecisionQuery> 或 <XACMLPolicyQuery> 应被封装在一个可能已签名的 <samlp:RequestAbstractType> 元素中。

<samlp:RequestAbstractType>的大部分成员在 ITU-T X.1141 建议书中都已详细说明。为了与本轮廓中定义并使用的 SAML 查询类型一起使用，元素<saml:Issuer>和元素<ds:Signature>应按照前面各节中说明的方式使用。除去在此说明之外，本轮廓不对<samlp:RequestAbstractType>元素中的信息加以要求或限制。

10.5.1 元素<saml:Issuer>

参见 10.4.1，元素<saml:Issuer>。

10.5.2 元素<ds:Signature>

参见 10.4.2，元素<ds:Signature>。

10.6 元素<samlp:Response>

<XACMLAuthzDecisionStatement> 或 <XACMLPolicyStatement> 应被封装在一个可能已签名的 <samlp:Response> 元素中。

<samlp:Response>的大部分成员在 ITU-T X.1141 建议书中都已详细说明。下列元素和 XML 属性在此进一步加以说明，以便与本轮廓中定义并使用的 SAML 声明类型一起使用。除去在此说明之外，本轮廓不对 <samlp:Response>元素中的信息加以要求或限制。

10.6.1 元素<samlp:Issuer>

参见 10.4.1，元素<saml:Issuer>。

10.6.2 元素<ds:Signature>

参见 10.4.2，元素<ds:Signature>。

10.6.3 元素<samlp:StatusCode>

<samlp:StatusCode> 元素是在<samlp:Response>中的<samlp:Status>元素的一个成员。

10.6.3.1 对<XACMLAuthzDecisionQuery>的应答

在对 <XACMLAuthzDecisionQuery> 请求的应答中，<samlp:StatusCode> Value XML 属性应取决于授权决定<xacml:Status>元素的<xacml:StatusCode>元素，如下所示：

- 1) urn:oasis:names:tc:SAML:2.0:status:Success
该 <samlp:StatusCode> Value XML 属性值当且仅当 <xacml:StatusCode> 值是 urn:oasis:names:tc:xacml:1.0:status:ok时使用。
- 2) urn:oasis:names:tc:SAML:2.0:status:Requester
该 <samlp:StatusCode> Value XML 属性值应在以下两种情况下使用：一种是当 <xacml:StatusCode> 的值为 urn:oasis:names:tc:xacml:1.0:status:missing-attribute，一种是由于在<xacml:Request>中出现语法错误，导致<xacml:StatusCode> 的值为 urn:oasis:names:tc:xacml:1.0:status:syntax-error。
- 3) urn:oasis:names:tc:SAML:2.0:status:Responder
该<samlp:StatusCode> Value XML 属性值在以下情况下使用：由于在<xacml:Policy>中或 <xacml:PolicySet> 中出现语法错误，导致 <xacml:StatusCode> 的值为 urn:oasis:names:tc:xacml:1.0:status:syntax-error。注意不是所有策略中的语法错误都能和给定查询的处理一起被检测出来，因此不是所有的策略语法错误都能以这种方式报告。
- 4) urn:oasis:names:tc:SAML:2.0:status:VersionMismatch
该 <samlp:StatusCode> Value XML 属性值仅当位于PDP的SAML接口不支持用于查询的SAML 请求消息的版本时使用。

10.6.3.2 对< XACMLPolicyQuery >的应答

在对<XACMLPolicyQuery> 请求的应答中, <samlp:StatusCode> 值 XML 属性应遵循 ITU-T X.1141 建议书中的说明。

11 XML数字签名轮廓

本节提出了一种为 XACML 模式实例提供认证和完整性保护的轮廓, 此轮廓与 W3C Signature:2002 一起使用。

仅仅当签名信息包含签名者身份和有效期(签名数据对象被认为有效的的时间)的明确说明时, 数字签名对认证和完整性保护才是有用的。XACML 自身并未定义这一信息的格式, 因为除了对接入控制策略、请求和应答的明确说明外, XACML 打算使用别的标准。

SAML 已经定义了一种适当的格式。第 10 节定义了一个将 SAML 和 XACML 模式实例一起使用的轮廓。因此这一轮廓推荐在 SAML 声明、请求和应答中使用 XACML 模式实例, 而这将随后被数字签名, 如同 ITU-T X.1141 建议书中所明确的。

11.1 SAML的使用

如同第 10 节中描述的那样, 本轮廓推荐在 SAML 声明、请求和应答中嵌入 XACML 模式实例的使用。这样的 SAML 对象将被数字签名, 如同 8.4/X.1141, SAML 和 XML 签名句法和处理中所描述的。

11.2 规范化

为了得到一个被可信赖的组织所证实的数字签名, 被签名的比特流必须与所证实的比特流完全一致。为了确保实现, 必须将 XML 签名文档规范化(见 W3C Canonicalization:2002)。ITU-T X.1141 建议书明确了专有规范化的使用(见 W3C Canonicalization:2002)。

11.2.1 XACML数据对象中的命名空间元素

任何将被签名的 XACML 数据对象必须明确说明在其中所使用的所有命名空间元素。如果不这样, 那么此数据对象将会引起从原型数据对象的命名空间的解析, 而诸多原型数据对象各有不同。

当专有规范被当作规范使用或者用作转换方法时, 则 XACML 数据对象中元素所使用的 XACML 模式的命名空间必须结合前缀并且包含在参数 InclusiveNamespacesPrefixList 之中(见 W3C Canonicalization:2002)。

11.2.2 补充的规范化考虑

为了确保被签名的数据对象与所证实的数据对象匹配, 通常必须执行 XACML 数据对象的补充转换。此处列出了这些转换中的一部分, 但是本轮廓并不旨在明确执行这些转换的算法。

如果一个 XACML 数据对象包含可能会有不止一种形式(如(TRUE, FALSE), (1, 0), (true, false)), 表示的数据元素, 那么必须定义且明确一种正常化这些数据元素的转换方法。

不论是在 XML 属性值中还是在 XACML 属性值中, 本轮廓推荐在相应的数据类型值之上应用下列规范。

- 1) 在<http://www.w3.org/2001/XMLSchema>模式定义之中定义了对XACML所定义的数据类型的规范表示法, 且数据类型的值必须符合<http://www.w3.org/2001/XMLSchema>模式定义中所明确的规范格式。这包括布尔{"true", "false"}, double, dateTime, time, date和hexBinary (大写)。
- 2) <http://www.w3.org/2001/XMLSchema#anyURI> — 使用IETF RFC 2396之中所定义的规范格式。
- 3) <http://www.w3.org/2001/XMLSchema#base64Binary> — 移除了所有的换行符和空格。移除了第一个“=”字符后面的所有字符。Base64转换(标识符: <http://www.w3.org/TR/xmlsig-core/#sec-Base-64>)在执行这一规范中适用。

- 4) urn:oasis:names:tc:xacml:1.0:data-type:x500Name — 首次正常化依据 IETF RFC 2253。如果有任何 RDN 包含多个 attributeTypeAndValue 对，当与八位组串比较时，按升序重定向此 RDN 中的 AttributeValuePairs（见 11.6/X.690）。
- 5) urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name — 将名称的域部分正常化为小写体。
- 6) XPath 表述 — <http://www.w3.org/2002/06/xmlsig-filter2> 用于将 XPath 表述转为规范格式。

11.3 签名模式

对任何 XACML 数据对象的分析都基于得到此 XACML 数据对象所依赖的所有模式的一个准确拷贝。需要注意的是，在 XACML 模式实例属性中所包含的模式 URI 并不能保证使用模式的一个准确拷贝：攻击者可能会将其替换成一个包含正确标识符的假的模式。签名可以起到保护 XACML 数据对象所依赖的模式不被替换和修改的作用。本节所讲述的就是为此目的的签名的使用。

在多数情况下，一个数据对象签名者必须在 <SignedInfo> 元素中包含对应 XACML 数据对象所依赖的每一模式的 <Reference> 元素，<SignedInfo> 元素含有包含 XACML 数据对象自身的 <Reference>。

在某些情况下，数据对象签名者确知，所有的 PDP 需要分析数据对象来赋值已知的 XACML 数据对象且得到特定模式的准确拷贝，并且不会要求 PDP 去核实此模式的摘要信息。在这种情况下，数据对象签名者可能会忽略一些不需要核实信息的模式的 <Reference> 元素。

12 XACML 的层次资源轮廓

通常情况下，资源是被组织成层次结构的。比如文件系统、XML 文档和体制。本节明确了 XACML 如何为被组织成层次结构的资源提供接入控制。

为什么资源组织成层次结构具备特殊性？首先，层次结构的策略常常是对层次结构的整个子树应用同样的接入控制。尽量表述一个统一的应用于层次结构各节点的整个子树的策略性约束，而不是为每一个节点的子树指定一个单独的约束，以增加使用的简易性和此策略能够准确反映期望接入控制的可能性。层次结构的资源的另一个特性是对某一节点的存取可能依赖于另一节点的属性值。比如，一个医疗病人只有在他的名字与节点“患者姓名”的值相匹配时，才被允许对 XML 文档医疗记录的节点“诊断”进行存取。这种情况中，被请求的节点不能被独立于层次结构中其它的节点而处理，并且 PDP 必须存取其它节点的值。最后，层次结构中节点的标识符常常取决于其在层次结构中的位置，而且可能会有多种方式来描述一个单个节点的标识符。为了能够顺利将此策略应用到各节点之上，必须注意将节点标识符的表述统一化。否则，通过请求一个其标识符与此策略所使用的标识符不同的节点，请求者可以绕过接入控制。

将资源组织成的层次结构可以是“树”（层次结构只有一个根节点）或者“森林”（层次结构有多个根节点），但是层次结构不能含有环路。这两种类型层次结构的另一种术语叫做“Directed Acyclic Graph”和“DAG”。本轮廓中称所有这样的资源为层次结构资源。一个 XML 文档资源的结构通常是一个“tree”。其它类型的层次结构资源的结构可能是“森林”，比如支持链接的文件系统中的文件。

层次结构资源中的节点被当作是彼此独立的资源。针对一个内部节点的允许存取的授权决定并不表示允许对其子节点的存取。而针对一个内部节点的拒绝存取的授权决定也同样不表示拒绝对其子节点的接入。

本轮廓明确了三种类型的设置来处理层次结构资源：

- 表示节点的身份
- 请求对节点的接入
- 确定适用于一个或多个节点的策略

对每种设置的支持都是可选的。

本节准备了两种方式表示层次结构资源。在第一种方式中，节点是其一部分的层次结构被表示成一个包含在请求之内的 XML 文档，并且被请求的资源被表示为此文档中的节点。在第二种方式中，被请求的资源并不被表示成 XML 文档中的节点，并且也没有请求之中层次结构部分的表示法。需要注意的是，第一种情况中的实际目标资源并不需要是 XML 文档的一部分——它仅仅是在请求中以那种方式表示。同样地，第二种情况中的目标资源可能实际上却是 XML 文档的一部分，但是在请求中却以其它的方式表示。因此，在请求中所表示的资源的结构和即将被接入的物理资源的实际结构之间并没有假定的相互关系。

处理在 XML 文档中被表示成节点的资源的设置可以利用 XML 文档自身被包含在决定的请求之中的既成事实。可以使用 XPath 表达式来以一种标准的方式索引文档中的节点，并且为文档中给出的节点提供唯一的表示法。如果层次结构资源不是 XML 文档，那么就不能使用这些设置。需要为这些未被表示为 XML 资源提供一些其它的方法来确定层次结构中被请求节点的位置。在有些情况下，这可通过将节点在层次结构中的位置包含到节点的身份信息之中来实现。在另外一些情况下，一个节点可能拥有不止一个标准的身份，比如当文件系统中文件的路径名包含有硬连接。在这种情况下，XACML PDP 的文本操纵者需要为所有节点的源提供身份信息。正因为这些原因，处理 XML 文档资源中节点的设置和处理其它层次结构资源中节点的设置不同。

处理层次结构资源时，在一个单独的决定请求中为资源中的多个节点请求授权决定是很有用的。本节可被看作是分层次的堆砌于多个资源轮廓之上（见第 9 节），它依次被分层次的堆砌在第 7 节所明确的动作之上。然而，本节的功能性被直接堆砌到第 7 节中的功能性之上。

针对层次结构资源，本节假设对层次结构资源中多个节点的接入请求都被分解为对单个节点的单独接入请求。

12.1 表示节点的身份信息

为了使 XACML 的策略统一适用于层次结构资源的节点，必须将此资源中的节点用一种统一的方式表示。如果策略提到节点使用一种表示法，但是请求提到节点使用另一种不同的表示法，那么此策略将不再适用，安全性也会减弱。

下面的各节描述了针对层次结构资源中节点的推荐表示法。只要关于此资源的所有策略管理问题和所有策略实施问题都规定使用此二选一表示法，那么已知资源中的节点的二选一表示法就是被允许的。

12.1.1 XML 文档资源中的节点

本部分是标准化的，但是可选的。

下面的 URI 被用作在本轮廓的这一部分所明确的功能性的标识符：

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-id
```

被表示成 XML 文档实例的资源的节点标识符是一个 XPath 表达式，此 XPath 表达式确切赋值了资源拷贝中的那个节点，并且被包含在 <Request> 的 <Resource> 元素的 <ResourceContent> 元素之中。

12.1.2 非 XML 文档资源中的节点

本部分是标准化的，但是可选的。

下面的 URI 被用作在本轮廓的这一部分所明确的功能性的标识符：

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-id
```

没有被表示成 XML 文档实例的层次结构资源中的节点的身份信息将会被表示成符合 IETF RFC 2396 的一个 URI。比如下面形式的 URI。

```
<scheme> ":" <authority> "/" <pathname>
```

文件系统资源必须使用 "file:" 模式。如果在 IETF RFC 2396 中或者在注册 URI 模式的相关标准之中没有为资源类型明确标准的 <scheme> 模式，那么 URI 必须使用 "file:" 模式。

URI 的 <pathname> 部分必须是如下形式：

```
<root name> [ "/" <node name> ]*
```

<root name>和<node name>属性值的序列必须与相应节点的各自层次结构组成部分的名称相符合，而此相应节点就是从<root>节点到被表示的节点的路径之上的节点的源节点。

必须满足下面的规则要求。

- URI的编码使用UTF8。
- URI的不区分大小写的部分使用小写体。
- 字符的忽略必须符合IETF RFC 2396。
- URI的<authority>部分必须明确并且必须是已知资源类型的标准权威表示法。这个<authority>可以使用域名系统（DNS）名称或者数字IPv4或IPv6地址来确定，其中DNS名称必须使用。
- URI的<pathname>部分的组成部分可使用<authority>处路径构成的标准形式来明确。
- 依照IETF RFC 2396，URI的<pathname>部分的层次结构组成部分之间的分隔符使用字符“/”。字符“/”的序列被分解为单独的“/”。节点身份信息不以字符“/”为结束。
- <pathname>不包含软连接。
- 所有<pathname>属性值都是绝对路径值。
- 从<authority>处的<root>到被表示的节点的绝对路径，如果有不止一个被充分的解决，那么在每一个这样的路径中的请求文本中必须出现独立的带有AttributeId "urn:oasis:names:tc:xacml:1.0:resource:resource-id"和DataType http://urn:oasis:names:tc:xacml:1.0:data-type:anyURI资源属性。

12.2 请求对节点的接入

为了使 XACML 的策略统一适用于层次结构资源的节点，必须将请求对此资源中的节点进行接入的请求文本用一种统一的节点接入描述法表示。如果策略涉及到节点的某种预期属性，但是请求文本不包括这些属性，或者属性没有以预期的方式表示，那么此策略将不再适用，安全性也会减弱。

下面的各节描述了针对层次结构资源中节点的推荐请求文本表示法。只要关于此资源的所有策略管理问题和所有策略实施问题都规定使用此二选一表示法，那么这些请求的二选一表示法就是被允许的。

12.2.1 XML文档资源中的节点

本部分内容是标准的，但是可选的。

下面的 URI 将被用作为在本轮廓中的这一部分所明确的功能性的标识符：

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req
```

带有 AttributeIds:

```
"urn:oasis::names:tc:xacml:2.0:resource:resource-parent"  
"urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor"
```

和:

```
"urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor-or-self"
```

的属性对设置来说是可选的。如果要支持用在 XML 文档资源之中，下面的 URI 将被用作为它们所代表的功能性的标识符：

```
"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-parent"  
"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-ancestor"
```

和:

```
"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-ancestor-or-self"
```

为了对表示为 XML 文档中节点的资源请求接入,请求文本<Resource>元素必须包含下面的元素和 XML 属性。

- 包含了整个XML文档实例的<ResourceContent>元素,被请求的节点是此XML文档实例的一部分。
- 带有AttributeId属性"urn:oasis::names:tc:xacml:1.0:resource:resource-id"和DataType属性"urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression"的<Attribute>元素。此<Attribute>的<AttributeValue>是一个XPath表达式,它的文本节点必须是<ResourceContent>元素的唯一子节点。此XPath表达式要赋值节点设置,将被请求存取的单独节点包含在<ResourceContent>元素之中。
- 带有AttributeId属性"urn:oasis::names:tc:xacml:2.0:resource:resource-parent"和DataType属性"urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression"的<Attribute>元素。此<Attribute>的<AttributeValue>是一个XPath表达式;这个XPath表达式的文本节点必须是<ResourceContent>元素的唯一子节点。此XPath表达式要赋值节点设置,将"resource-id"属性中所表示节点的直接父节点这一单独节点包含在<ResourceContent>元素之中。
- 涉及在XML文档实例之中并且是"resource-id"属性所代表节点的源的每一个节点,元素<Attribute>带有AttributeId属性"urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor"和DataType属性"urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression"。此<Attribute>的<AttributeValue>必须是一个XPath表达式;且其文本节点必须是<ResourceContent>元素的唯一子节点。此XPath表达式要赋值节点设置,将"resource-id"属性中所表示节点的各自的源节点包含在<ResourceContent>元素之中。对于每一个"resource-parent"属性应当有一个相应的"resource-ancestor"属性。此<Attribute>将会明确一个Issuer。
- 涉及在XML文档实例之中并且是"resource-id"属性所代表节点的源的每一个节点和"resource-id"节点自身,元素<Attribute>带有AttributeId属性"urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor-or-self"和DataType属性"urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression"。此<Attribute>的<AttributeValue>必须是一个XPath表达式;且其文本节点必须是<ResourceContent>元素的唯一子节点。此XPath表达式要赋值节点设置,将"resource-id"属性中所表示节点的各自的源节点或者是"resource-id"节点自身包含在<ResourceContent>元素之中。对每一个"resource-parent"和"resource-id"属性应当有一个相应的"resource-ancestor-or-self"属性。此<Attribute>属性将会明确一个Issuer。

<Resource>元素还会包含有额外的属性。特别是,可能会包含有下面的属性:

- 带有AttributeId属性"urn:oasis::names:tc:xacml:2.0:resource:document-id"和DataType属性"urn:oasis:names:tc:xacml:1.0:data-type:anyURI"的<Attribute>元素。此<Attribute>的<AttributeValue>必须是标识XML文档的一个URI,此处的XML文档包含被请求的资源且它的一个拷贝存在于<ResourceContent>元素之中。此<Attribute>会明确一个Issuer。

12.2.2 非XML文档资源中的节点

本部分内容是标准的,但是可选的。

下面的URI被用作是在本段中的这一部分所明确的功能性的标识符:

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req
```

带有 AttributeIds:

```
"urn:oasis::names:tc:xacml:2.0:resource:resource-parent"
```

```
"urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor"
```

和:

```
"urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor-or-self"
```

的属性对设置来说是可选的。如果要支持用在未被表示为 XML 文档的资源之中，下面的 URI 将被用作它们所代表的功能性的标识符:

```
"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-parent"  
"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-ancestor"
```

和:

```
"urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-ancestor-or-self"
```

为了对一个未被表示为 XML 文档的层次结构资源中的节点请求接入，请求文本<Resource>元素不能包含<ResourceContent>元素。请求文本<Resource>元素必须包含下面的元素和 XML 属性。需要注意的是未被表示为 XML 文档的层次结构资源中的节点可以有多个父节点。比如，在一个支持硬连接的文件系统中，一个单独的节点就可能会有多个标准路径。每一个这样的路径含有一组不同的父节点和源。

- 涉及到被请求节点的每一个标准表示法，<Attribute>元素带有 AttributeId 属性 "urn:oasis::names:tc:xacml:1.0:resource:resource-id"。此<Attribute>的<AttributeValue>必定是被请求接入的节点的唯一的标准身份信息。此<Attribute>的 DataType 取决于为此特定资源中节点的标识符所选择的表示法。
- 涉及到在 "resource-id" 属性或属性簇中所明确节点的每一个直接父节点和其每一个标准表示法，<Attribute>元素带有 AttributeId 属性 "urn:oasis::names:tc:xacml:2.0:resource:resource-parent"。此<Attribute>的<AttributeValue>必定是此父节点的标准身份信息。此<Attribute>的 DataType 必取决于为此特定资源中节点的标识符所选择的表示法。此<Attribute>会明确一个 Issuer。如果被请求的节点是一个森林的一部分而不是一个单独的树的一部分，或者其父节点有不止一个标准表示法，这里应该有至少一个此属性的实例，对应去往派生被请求节点的根节点的路径之上的每个父节点，以及每个这样的父节点的每个标准表示法。
- 涉及到在 "resource-id" 属性或属性簇中所明确节点的每一个源节点和其每一个标准表示法，<Attribute>元素带有 AttributeId 属性 "urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor"。此<Attribute>的<AttributeValue>必定是此源节点的标准身份信息。此<Attribute>的 DataType 必取决于为此特定资源中节点的标识符所选择的表示法。此<Attribute>会明确一个 Issuer。针对每一个 "resource-parent" 属性都会有一个相应的 "resource-ancestor" 属性。如果被请求的节点是一个森林的一部分而不是一个单独的树的一部分，或者其源节点有不止一个标准表示法，这里应该有至少一个此属性的实例，对应去往派生被请求节点的根节点的路径之上的每个源节点，以及每个这样的源节点的每个标准表示法。此属性值并不一定要反映每个源节点在层次结构中的位置。
- 涉及到在 "resource-id" 属性或属性簇中所明确节点的每一个源节点和其每一个标准表示法，以及 "resource-id" 节点自身的标准表示法，<Attribute>元素带有 AttributeId 属性 "urn:oasis::names:tc:xacml:2.0:resource:resource-ancestor-or-self"。此<Attribute>的<AttributeValue>必定是源节点或 "resource-id" 节点自身的单独身份信息。此<Attribute>的 DataType 必取决于为此特定资源中节点的标识符所选择的表示法。此<Attribute>会明确一个 Issuer。针对每一个 "resource-ancestor" 和 "resource-id" 属性都会有一个相应的 "resource-ancestor-or-self" 属性。如果被请求的节点是一个森林的一部分而不是一个单独的树的一部分，或者其源节点有不止一个标准表示法，这里应该有至少一个此属性的实例，对应去往派生被请求节点的根节点的路径之上的每个源节点，以及每个这样的源节点的每个标准表示法。此属性值并不一定要反映每个源节点在层次结构中的位置。

额外的属性包含在<Resource>元素之中。

12.3 确定节点所适用的策略

本节是资料性的。

本节描述了多种方式来明确适用于层次结构资源中多个节点的策略。这并不旨在成为一个详尽的目录。

12.3.1 适用于任何层次结构资源节点的策略

本节是资料性的。

如 12.5 中所描述的,具有如下属性质的资源属性可以用来确定适用于任何层次结构中一个或多个节点的策略。

```
urn:oasis:names:tc:xacml:2.0:resource:resource-parent  
urn:oasis:names:tc:xacml:2.0:resource:resource-ancestor  
urn:oasis:names:tc:xacml:2.0:resource:resource-ancestor-or-self
```

需要注意的是,涉及到"resource-parent"、"resource-ancestor"或"resource-ancestor-or-self"属性的<ResourceAttributeDesignator>会返回一些代表着所有的标准的标识符的属性值,这所有的标准标识符属于父节点、源或者资源自身的源,而存取被分别请求到资源。这些父节点、源或者资源自身的标识符的代表并不一定要表明从根节点到各自的父节点、源或者资源自身,除非使用了 12.2.2 中所推荐使用的代表,此时资源中的节点不是一个 XML 文档。

标准的 XACML 包和高阶包函数可用来确定适用于任何层次结构中的一个或多个节点的策略。作为这些函数的参数的节点会可以使用具有"resource-parent"、"resource-ancestor"或"resource-ancestor-or-self"属性值的<ResourceAttributeDesignator>来规定。

12.3.2 仅适用于XML文档节点的策略

本节是资料性的。

对于 XML 文档实例的层次结构资源,下面的函数可用于确定适用于此资源中一个或多个节点的策略。

```
urn:oasis:names:tc:xacml:2.0:function:xpath-node-match
```

标准的 XACML 元素<AttributeSelector>可用在策略之中来接入所有或者部分代表了 XML 文档的资源,并且此元素包含在表示请求文本的元素<ResourceContent>之中。

标准的 XACML 包和高阶包函数可用来确定适用于任何层次结构中的一个或多个节点的策略。用为这些函数的参数的节点会可以使用选择了<Resource>元素中的一部分<ResourceContent>元素的<AttributeSelector>来规定。

12.3.3 仅适用于非XML资源的策略

本节是资料性的。

对于未被表示为 XML 文档实例的层次结构资源,并且使用了本轮廓中所明确的节点的 URI 表示法,下面的函数可用于确定适用于此资源中一个或多个节点的策略。

```
urn:oasis:names:tc:xacml:1.0:function:anyURI-equal  
urn:oasis:names:tc:xacml:1.0:function:regexp-uri-match
```

12.4 新DataType: xpath-expression

本部分是标准的,但是可选的。

下面的 XML DataType 属性值可与 XML 文档层次结构资源配套使用。为了支持第 12.1.1 必须支持此 DataType。

下面的 URI 所表示的 `DataType` 的值是 XPath 表达式。具有此 `DataType` 的属性值应作为 XPath 表达式来解释的字符串。赋值此属性的结果将是由赋值 XPath 表达式的结果的节点集。如果此字符串不是一个有效的 XPath 表达式，赋值此属性的结果将是 "Indeterminate"。

```
urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression
```

12.5 新属性标识符

本部分是标准的，但是可选的。

12.5.1 文档-id

下面的标识符表明了 XML 文档的身份，此 XML 文档层次结构资源包括被请求的资源并且其 `<ResourceContent>` 元素中含有一个拷贝。不管什么时候请求对 XML 文档资源中节点的接入，请求文本的 `<Resource>` 元素中会提供一个或多个带有 `AttributeId` 的属性实例。这些属性的 `DataType` 应该是 "urn:oasis:names:tc:xacml:1.0:data-type:anyURI"。

```
urn:oasis:names:tc:xacml:2.0:resource:document-id
```

12.5.2 资源的父（节点）

下面的标识符表明了被请求节点所属资源树或森林中的一个父节点的标准身份。不管什么时候请求对 XML 文档资源中节点的接入，请求文本的 `<Resource>` 元素中都会为每一个被请求节点的父节点的每一个标准表示法提供一个带有 `AttributeId` 的属性实例。

```
urn:oasis:names:tc:xacml:2.0:resource:resource-parent
```

12.5.3 资源的祖先

下面的标识符表明了被请求节点所属资源树或森林中的一个源节点的标准身份。不管什么时候请求对 XML 文档资源中节点的接入，请求文本的 `<Resource>` 元素中都会为每一个被请求节点的祖先节点的每一个标准表示法提供一个带有 `AttributeId` 的属性实例。

```
urn:oasis:names:tc:xacml:2.0:resource:resource-ancestor
```

12.5.4 资源的祖先或它自身

下面的标识符表明了被请求节点所属资源树或森林中的一个源节点的标准身份，或者被请求节点自身的标准身份。不管什么时候请求对 XML 文档资源中节点的接入，请求文本的 `<Resource>` 元素中都会为每一个被请求节点的祖先节点的每一个标准表示法，或者是被请求节点自身的每一个标准表示法提供一个带有 `AttributeId` 的属性实例。

```
urn:oasis:names:tc:xacml:2.0:resource:resource-ancestor-or-self
```

12.6 新的轮廓标识符

下面的 URI 值用作本轮廓中各节所明确的功能性的标识符：

第 12.1.1 节：XML 文档资源节点

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-id
```

第 12.1.2 节：非 XML 文档资源节点

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-id
```

第 12.2.1 节：在一个 XML 文档资源中的节点

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req
```

对 "resource-parent"、"resource-ancestor" 和 "resource-ancestor-or-self" 属性的支持在本节中是可选的，所以此处有不同的标识符：

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-parent
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-ancestor
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:xml-node-req:resource-ancestor-or-self
```

第 12.2.2 节：在一个非 XML 文档资源中的节点

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req
```

对"resource-parent"、"resource-ancestor"和"resource-ancestor-or-self"属性的支持在本节中是可选的，所以此处有不同的标识符：

```
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-parent
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-ancestor
urn:oasis:names:tc:xacml:2.0:profile:hierarchical:non-xml-node-req:resource-ancestor-or-self
```

13 保密策略轮廓

数据管理员的两个职责是确保个人数据的使用被限制在搜集它的目的的履行之上或者是其它与之不相容的目的之上，或者是除了数据主体或者通过法律局的同意外，防止个人数据的败露。为了强化这些职责，本节为标准属性和标准<Rule>元素提供了轮廓，这与私有可识别信息被搜集和使用的目的有关。

13.1 标准属性

本规则定义了两个属性。

```
urn:oasis:names:tc:xacml:2.0:resource:purpose
```

类型为“http://www.w3.org/2001/XMLSchema#string”的这一属性明确了数据资源被搜集的目的。必须告知资源的所有者并且资源所有者同意对其资源的这一目的的使用。属性值应该是规则的表述。管理员的保密策略必须定义所有可取值的语义。

```
urn:oasis:names:tc:xacml:2.0:action:purpose
```

类型为“http://www.w3.org/2001/XMLSchema#string”的这一属性明确了对数据资源接入请求的目的。动作的目的可以是层次组织的，在这种情况下，属性值必须表示层次结构中的一个节点。

13.2 标准规则：匹配目的

这一规则必须与"urn:oasis:names:tc:xacml:2.0:rule-combining-algorithm:deny-overrides"规则组合算法一起使用。它规定接入必须被拒绝，除非根据规定的匹配规则，所请求接入的目的与数据资源被搜集的目的相匹配。

```
<?xml version="1.0" encoding="UTF-8"?>
<Rule xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os" RuleId="
urn:oasis:names:tc:xacml:2.0:matching-purpose"
Effect="Permit">
  <Condition FunctionId="urn:oasis:names:tc:xacml:2.0:function:regexp-
string-match">
    <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:2.0:resource:purpose"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
    <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:2.0:action:purpose"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
  </Condition>
</Rule>
```

附件 A

数据类型和函数

A.1 引言

本附件规定 XACML 中用于创建条件和目标匹配的声明的数据类型和函数。

本建议书描述了原语数据类型和包。命名了标准函数并描述了其操作语义。

注一 数字值的信息字符串表示参见 [IEEE 754] 和 [RBAC]。

A.2 数据类型

虽然 XML 示例用字符串表示所有数据类型，在它们有字符串表示时，XACML PDP 必须解释数据类型不仅是字符串。字符串，布尔值，整数和双精度类型必须从它们的 XML 字符串表示转化为值以使其可与和它们的讨论域内的其它值相比较，例如数字。以下原语数据类型指定为与 XACML 一起使用并且有明确的数据表示：

- <http://www.w3.org/2001/XMLSchema#string>
- <http://www.w3.org/2001/XMLSchema#boolean>
- <http://www.w3.org/2001/XMLSchema#integer>
- <http://www.w3.org/2001/XMLSchema#double>
- <http://www.w3.org/2001/XMLSchema#time>
- <http://www.w3.org/2001/XMLSchema#date>
- <http://www.w3.org/2001/XMLSchema#dateTime>
- <http://www.w3.org/2001/XMLSchema#anyURI>
- <http://www.w3.org/2001/XMLSchema#hexBinary>
- <http://www.w3.org/2001/XMLSchema#base64Binary>
- <urn:oasis:names:tc:xacml:2.0:data-type:dayTimeDuration>
- <urn:oasis:names:tc:xacml:2.0:data-type:yearMonthDuration>
- <urn:oasis:names:tc:xacml:1.0:data-type:x500Name>
- <urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name>
- <urn:oasis:names:tc:xacml:2.0:data-type:ipAddress>
- <urn:oasis:names:tc:xacml:2.0:data-type:dnsName>

为了增强互用性，建议所有时间基准是 UTC 时间。

XACML PDP 必须能把字符串表示转换成各种各样的原语数据类型。

注一 对于整数和双精度类型，XACML 将使用 [IEEE 754] 中描述的转换方法。

XACML 定义了 6 种数据类型，它们是：

```
"urn:oasis:names:tc:xacml:2.0:data-type:dayTimeDuration"  
"urn:oasis:names:tc:xacml:2.0:data-type:yearMonthDuration"  
"urn:oasis:names:tc:xacml:1.0:data-type:x500Name"  
"urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"  
"urn:oasis:names:tc:xacml:2.0:data-type:ipAddress"  
"urn:oasis:names:tc:xacml:2.0:data-type:dnsName"
```

这些类型代表主体或资源标识符并且出现于一些标准应用中，例如 TLS/SSL 和电子邮件。

A.2.1 以日期和时间表示的期限

"urn:oasis:names:tc:xacml:2.0:data-type:dayTimeDuration" 原语类型被定义为 **xs:duration** 类型的限制，它仅保留了符号、年和月构件。


```
<xs:simpleType name='urn:oasis:names:tc:xacml:2.0:data-type:yearMonthDuration'>
  <xs:restriction base='xs:duration'>
    <xsd:pattern value="[-]?P\p{Nd}+(Y(\p{Nd}+M)?|M)"/>
  </xs:restriction>
</xs:simpleType>
```

yearMonthDuration 的值以月为单位，即：

```
(‘年构件的值’*12) + (‘月构件的值’)
```

如果符号构件为“-”，则结果值为负；否则，结果值为正。

A.2.2 以年和月表示的期限

"urn:oasis:names:tc:xacml:2.0:data-type:yearMonthDuration"原语类型被定义为 **xs:duration** 类型的限制，它仅保留了符号、日、小时、分和秒构件。

```
<xs:simpleType name='urn:oasis:names:tc:xacml:2.0:data-type:dayTimeDuration'>
  <xs:restriction base='xs:duration'>
    <xsd:pattern value="[-]?P(\p{Nd}D(T(\p{Nd}+(H(\p{Nd}+(M(\p{Nd}+(\.\p{Nd}*)?S|\.\p{Nd}+S)?|(\.\p{Nd}*)?S)|(\.\p{Nd}*)?S)?|M(\p{Nd}+(\.\p{Nd}*)?S|\.\p{Nd}+S)?|T(\p{Nd}+(H(\p{Nd}+(M(\p{Nd}+(\.\p{Nd}*)?S|\.\p{Nd}+S)?|(\.\p{Nd}*)?S)|(\.\p{Nd}*)?S)?|M(\p{Nd}+(\.\p{Nd}*)?S|\.\p{Nd}+S)?|(\.\p{Nd}*)?S)|(\.\p{Nd}*)?S)|M(\p{Nd}+(\.\p{Nd}*)?S|\.\p{Nd}+S)?|(\.\p{Nd}*)?S)|(\.\p{Nd}+S)))/>
  </xs:restriction>
</xs:simpleType>
```

dayTimeDuration 的值以秒为单位，并且有：

```
(‘日构件的值’*24) +
(‘小时构件的值’*60) +
(‘分钟构件的值’*60) +
(‘秒构件的值’)
```

如果符号构件为“-”，则结果值为负，否则结果值为正。

A.2.3 X.500号码簿名称

"urn:oasis:names:tc:xacml:1.0:data-type:x500Name"原语类型代表了一个 X.520 专有名称。IETF RFC 2253 中描述了这种名称的有效语法。

A.2.4 IETF RFC 822 名称

"urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"原语类型代表了一个电子邮件地址。IETF RFC 2821, 4.1.2 中描述了这种名称的有效语法。

A.2.5 IP地址

"urn:oasis:names:tc:xacml:2.0:data-type:ipAddress"原语类型代表一个 IPv4 或 IPv6 网络地址，带有可选的掩码和可选的端口或端口范围。其语法应为：

```
ipAddress = address [ "/" mask ] [ ":" [ portrange ] ]
```

对于 IPv4 地址，其地址和掩码按照 IETF RFC 2396, 3.2 中"host"的语法进行格式化。

对于 IPv6 地址，其地址和掩码按照 IETF RFC 2732 中"ipv6reference"的语法格式化。（注：在此语法中，IPv6 地址或掩码以括号"[]"括起来）

A.2.6 DNS名称

"urn:oasis:names:tc:xacml:2.0:data-type:dnsName"原语类型代表一个域名系统（DNS）主机名，带有可选的端口或端口范围。其语法为：

```
dnsName = hostname [ ":" portrange ]
```

除了通配符“*”可以用于主机名的最左构件指示其右边域规定的域下的“任何子域”，主机名按照 IETF RFC 2396, 3.2 格式化。

对"urn:oasis:names:tc:xacml:2.0:data-type:ipAddress"和"urn:oasis:names:tc:xacml:2.0:data-type:dnsName"两种数据类型，其端口或端口范围语法为：

```
portrange = portnumber | "-"portnumber | portnumber "-" [portnumber]
```

其中"portnumber"为一十进制端口号。如果端口号形式为“-x”，其中"x"是一个端口号，则其范围是"x"及其以下的所有端口号。如果端口号形式为“x-”，则其范围是"x"及其以上的所有端口号。

A.3 函数

XACML 规定了以下函数。如果这些函数之一的参数被赋值为“Indeterminate”，那么此函数将被置为“Indeterminate”。

A.3.1 平等判断

以下函数是各种原语类型的等值函数。某个特定数据类型的每一个函数遵循那个数据类型指定的标准约定。

```
urn:oasis:names:tc:xacml:1.0:function:string-equal
```

这个函数应采用数据类型为"http://www.w3.org/2001/XMLSchema#string"的两个参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当函数的两个参数长度相等且按照"integer-equal"函数逐字节确定每个字符串都相等时，函数应返回"True"，否则应返回"False"。

```
urn:oasis:names:tc:xacml:1.0:function:boolean-equal
```

这个函数需要采用两个数据类型"http://www.w3.org/2001/XMLSchema#boolean"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当函数的两个参数相等时，函数应返回"True"，否则应返回"False"。

```
urn:oasis:names:tc:xacml:1.0:function:integer-equal
```

这个函数需要采用两个数据类型"http://www.w3.org/2001/XMLSchema#integer"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。

注 1 — 参见[IEEE 754]对于信息在整数上的整数计算。

```
urn:oasis:names:tc:xacml:1.0:function:double-equal
```

这个函数需要采用两个数据类型"http://www.w3.org/2001/XMLSchema#double"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。

注 2 — 如何赋值双精度值参见[IEEE 754]。

```
urn:oasis:names:tc:xacml:1.0:function:date-equal
```

这个函数需要采用两个数据类型"http://www.w3.org/2001/XMLSchema#date"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当函数的两个参数值相等时，函数应返回"True"。如果任一参数缺少一个明确时区，则时区值将由执行者提供。

```
urn:oasis:names:tc:xacml:1.0:function:time-equal
```

这个函数应采用两个数据类型"http://www.w3.org/2001/XMLSchema#time"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当函数的两个参数值相等时，函数应返回"True"。如果任一参数缺少一个明确时区，则时区值将由执行者提供。

```
urn:oasis:names:tc:xacml:1.0:function:dateTime-equal
```

这个函数应采用两个数据类型"http://www.w3.org/2001/XMLSchema#dateTime"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当函数的两个参数值相等时，函数应返回"True"。如果任一参数缺少一个明确时区，则时区值将由执行者提供。

```
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal
```

这个函数应采用两个数据类型"urn:oasis:names:tc:xacml:2.0:data-types:dayTimeDuration"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当函数的两个参数值相等时，函数应返回"True"。注意以词汇表示的每个参数必须被转化成以分数秒表示的值。

```
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal
```

这个函数应采用两个数据类型"urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当函数的两个参数值相等时，函数应返回"True"。注意以词汇表示的每个参数必须被转化成以整数月表示的值。

```
urn:oasis:names:tc:xacml:1.0:function:anyURI-equal
```

这个函数需要接受两个"http://www.w3.org/2001/XMLSchema#anyURI"数据类型的参数并返回一个"http://www.w3.org/2001/XMLSchema#boolean"类型。当且仅当函数的两个参数逐个码点比较都相等时，函数应返回"True"。

```
urn:oasis:names:tc:xacml:1.0:function:x500Name-equal
```

这个函数应采用两个数据类型"urn:oasis:names:tc:xacml:1.0:data-type:x500Name"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当两个参数中的相关区分名(RDN)值相匹配时，它应返回"True"。否则，应返回"False"。当且仅当下面的操作的结果是"True"，两个RDN被称为匹配。

- 1) 根据IETF RFC 2253规范化两个参数。
- 2) 如果任何RDN包含多个attributeTypeAndValue对，那么作为八位组串比较时（在11.6/X.690中描述），对该RDN中的AttributeValuePairs重新以升序进行排序。
- 3) 使用IETF RFC 3280中4.1.2.4节的规则来比较RDN。

```
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal
```

这个函数应采用两个数据类型"urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当这两个参数相等时，它应返回"True"，否则，它返回"False"。一个IETF RFC822名字由“本地名部分后随“@”，后随域名部分组成。本地名部分区分大小写，而域名部分（通常是一个DNS域名）不需区分大小写。执行下列操作：

- 1) 规范每个参数的域名部分到小写字母。
- 2) 通过将规范的参数应用于函数"urn:oasis:names:tc:xacml:1.0:function:string-equal"来比较表达式。

```
urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal
```

这个函数应采用两个数据"http://www.w3.org/2001/XMLSchema#hexBinary"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。如果由两个参数的值表示的八位组字节序列具有相同长度，并且使用"urn:oasis:names:tc:xacml:1.0:function:integer-equal"函数对逐个码点比较都相等，它应返回"True"，否则，它应返回"False"。从字符串表示法到八位组序列的变换参照W3C Datatypes:2001, 3.2.15中的规定。

```
urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal
```

这个函数应采用两个数据类型"http://www.w3.org/2001/XMLSchema#base64Binary"的参数并应返回一个"http://www.w3.org/2001/XMLSchema#boolean"。如果由两个参数的值表示的八位组序列具有相同长度，并且使用"urn:oasis:names:tc:xacml:1.0:function:integer-equal"函数对逐个码点比较都相等，应返回"True"，否则，应返回"False"。从字符串表示法到八位组序列的变换参照W3C Datatypes:2001, 3.2.16中的规定。

A.3.2 算术函数

下面所有的函数将采用两个指定数据类型为整数或双精度的参数，并应分别返回一个整数或双精度数据类型的元素。然而，“add”函数应带多于两个参数。在一个包含这些函数中任何一个的表达式中，如果任何参数是“Indeterminate”，那么该表达式应赋值为“Indeterminate”。在除法函数的情况下，如果除数为0，那么此函数应赋值为“Indeterminate”。

注一 每个函数赋值应该按由它们在[IEEE 754]中的逻辑副本指定的方式进行。

```
urn:oasis:names:tc:xacml:1.0:function:integer-add
```

该函数可能有两个或多个参数。

```
urn:oasis:names:tc:xacml:1.0:function:double-add
```

该函数可能有两个或多个参数。

```
urn:oasis:names:tc:xacml:1.0:function:integer-subtract  
urn:oasis:names:tc:xacml:1.0:function:double-subtract  
urn:oasis:names:tc:xacml:1.0:function:integer-multiply  
urn:oasis:names:tc:xacml:1.0:function:double-multiply  
urn:oasis:names:tc:xacml:1.0:function:integer-divide  
urn:oasis:names:tc:xacml:1.0:function:double-divide  
urn:oasis:names:tc:xacml:1.0:function:integer-mod
```

下面的函数应采用规定的数据类型单一参数。Round 和 floor 函数应带有单个数据类型为“http://www.w3.org/2001/XMLSchema#double”的参数，并返回一个数据类型为“http://www.w3.org/2001/XMLSchema#double”的值。

```
urn:oasis:names:tc:xacml:1.0:function:integer-abs  
urn:oasis:names:tc:xacml:1.0:function:double-abs  
urn:oasis:names:tc:xacml:1.0:function:round  
urn:oasis:names:tc:xacml:1.0:function:floor
```

A.3.3 序列转换函数

下面的函数在数据类型“http://www.w3.org/2001/XMLSchema#string”原语类型值之间转换。

```
urn:oasis:names:tc:xacml:1.0:function:string-normalize-space
```

此函数将带有一个数据类型为“http://www.w3.org/2001/XMLSchema#string”的参数并应通过去掉所有第一个和最后一个空格字符的方式来规范其值。

```
urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case
```

此函数将带有一个数据类型为“http://www.w3.org/2001/XMLSchema#string”的参数并应通过将大写字母转换为等同的小写字母的方式来规范其值。

A.3.4 数字的数据类型转换函数

下面的函数在数据类型“http://www.w3.org/2001/XMLSchema#integer”和原语类型“http://www.w3.org/2001/XMLSchema#double”之间转换。

```
urn:oasis:names:tc:xacml:1.0:function:double-to-integer
```

该函数应带有一个数据类型为“http://www.w3.org/2001/XMLSchema#double”的一个参数，并应将其数字值截短为一个完整数字，并返回一个数据类型为“http://www.w3.org/2001/XMLSchema#integer”的元素。

```
urn:oasis:names:tc:xacml:1.0:function:integer-to-double
```

此函数应带有一个数据类型为“http://www.w3.org/2001/XMLSchema#integer”的一个参数，并应用同样的数值将该函数值提高为数据类型为“http://www.w3.org/2001/XMLSchema#double”的元素。

A.3.5 逻辑函数

本节包括规定了对数据类型为“http://www.w3.org/2001/XMLSchema#boolean”的参数进行操作的逻辑函数的规范。

```
urn:oasis:names:tc:xacml:1.0:function:or
```

如果没有参数，此函数应返回“False”，如果至少有一个参数的赋值为“True”，那么应返回“True”。赋值顺序应从第一个参数到最后一个。如果任何参数赋值为“True”，不管其他未赋值的参数，那么该赋值将停止，并且结果为“True”。

```
urn:oasis:names:tc:xacml:1.0:function:and
```

如果没有参数，此函数应返回“True”，如果其参数中有一个赋值为“False”。那么该函数将返回“False”。赋值顺序将从第一个参数到最后一个。如果任何参数赋值为“False”，不管其他未赋值的参数，那么该赋值将停止，并且结果为“False”。

```
urn:oasis:names:tc:xacml:1.0:function:n-of
```

此函数的第一个参数应是数据类型 `http://www.w3.org/2001/XMLSchema#integer`。剩下的参数应是数据类型 `http://www.w3.org/2001/XMLSchema#boolean`。为使表达式为“True”，第一个参数规定了对要考虑为“True”的表达式赋值为“True”的其他参数的最小数目。如果第一个参数为 0，结果将为“True”。如果在第一个参数之后的参数数目少于第一个参数的值，那么该表达式的结果为“Indeterminate”。赋值顺序应该是：首先赋值整数值，然后赋值后续的所有参数。如果参数的规定数目赋值为“True”，那么赋值应停止并返回“True”。如果确定赋值剩余参数将不能满足需要，参数的赋值应停止。

```
urn:oasis:names:tc:xacml:1.0:function:not
```

此函数应带有一个数据类型为“`http://www.w3.org/2001/XMLSchema#boolean`”的一个参数。如果该参数赋值为“True”，那么此表达式的结果为“False”。如果参数赋值为“False”，那么该表达式的结果将为“True”。

注一 为确定该参数的赋值是否将导致“Indeterminate”，在赋值与、或或者n-of，试图对每一个参数进行所有赋值可能是不必要的。考虑其属性可用性的参数分析或其他有关错误的分析，例如被0除，会使得参数不出错误。在赋值被置为停止之后的某一点中的表达式中出现该类参数不必处理。

A.3.6 数字比较函数

这些函数形成一个用于比较两个数字的最小集，并产生一个布尔值结果。

注一 这些函数应遵守[IEEE 754]管理的规则。

```
urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal
urn:oasis:names:tc:xacml:1.0:function:integer-less-than
urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal
urn:oasis:names:tc:xacml:1.0:function:double-greater-than
urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal
urn:oasis:names:tc:xacml:1.0:function:double-less-than
urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal
```

A.3.7 日期和时间算术函数

这些函数对日期和时间进行算术操作。

```
urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration
```

这个函数应带有两个参数，第一个是数据类型“`http://www.w3.org/2001/XMLSchema#dateTime`”，第二个是数据类型“`urn:oasis:names:tc:xacml:2.0:data-types:dayTimeDuration`”。它应返回一个“`http://www.w3.org/2001/XMLSchema#dateTime`”的结果。该函数将通过根据 W3C DataTypes:2001 的附录 E 将第二个参数加入第一个参数来返回值。

```
urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration
```

这个函数将带有两个参数，第一个为“`http://www.w3.org/2001/XMLSchema#dateTime`”，第二个是“`urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration`”。它应返回一个“`http://www.w3.org/2001/XMLSchema#dateTime`”的结果。该函数应通过根据 W3C DataTypes:2001 的附录 E 将第二个参数加入到第一个参数来返回值。

```
urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration
```

这个函数应带有两个参数，第一个为“`http://www.w3.org/2001/XMLSchema#dateTime`”，第二个是“`urn:oasis:names:tc:xacml:2.0:data-types:dayTimeDuration`”。它将返回一个“`http://www.w3.org/2001/XMLSchema#dateTime`”的结果。如果第二个参数持续时间为正，该函数将根据 W3C DataTypes:2001 的附录 E 增加负的持续时间来返回值。如果第二个参数的持续时间为负，那么其返回结果就象将函数“`urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration`”应用于相应的正的持续时间。

urn:oasis:names:tc:xacml:1.0:function:date-time-subtract-yearMonthDuration

这个函数应带有两个参数，第一个为 "http://www.w3.org/2001/XMLSchema#dateTime"，第二个是 "urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration"。它将返回一个 "http://www.w3.org/2001/XMLSchema#dateTime" 的结果。如果第二个参数持续时间为正，该函数将根据 W3C DataTypes:2001 的附录 E 增加负的持续时间来返回值。如果第二个参数的持续时间为负，那么其返回结果就象将函数 "urn:oasis:names:tc:xacml:1.0:function:date-time-add-yearMonthDuration" 应用于相应的正的持续时间。

urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration

这个函数应带有两个参数，第一个为 "http://www.w3.org/2001/XMLSchema#date"，第二个是 "urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration"。它应返回一个 "http://www.w3.org/2001/XMLSchema#date" 类型的结果。该函数将通过根据 W3C DataTypes:2001 的附录 E 将第二个参数加入第一个参数来返回值。

urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration

这个函数应带有两个参数，第一个为 "http://www.w3.org/2001/XMLSchema#date"，第二个是 "urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration"。它应返回一个 "http://www.w3.org/2001/XMLSchema#date" 类型的结果。如果第二个参数持续时间为正，该函数将根据 W3C DataTypes:2001 的附录 E 增加负的持续时间来返回值。如果第二个参数持续时间为负，则其返回值就象将函数 "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" 应用于相应的正的持续时间。

A.3.8 非数字比较函数

这些函数对两个非数字类型的参数执行比较操作。

urn:oasis:names:tc:xacml:1.0:function:string-greater-than

此函数应带有两个数据类型为 "http://www.w3.org/2001/XMLSchema#string" 的参数，并应返回一个 "http://www.w3.org/2001/XMLSchema#boolean" 类型。当且仅当下面这种情况，此函数返回 "True"，其参数逐字节进行比较，并且在来自经 "urn:oasis:names:tc:xacml:1.0:function:integer-equal" 计算认为相等的两个参数的相应字节的初始前缀之后，下一个逐字节的比较就是使用函数 "urn:oasis:names:tc:xacml:2.0:function:integer-greater-then" 比较来自第一个参数的字节大于来自第二个参数的字节。否则它将返回 "False"。

urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal

此函数将带有两个数据类型为 "http://www.w3.org/2001/XMLSchema#string" 的参数，并返回一个 "http://www.w3.org/2001/XMLSchema#boolean"。它将返回一个结果，就好像通过带有包含函数 "urn:oasis:names:tc:xacml:1.0:function:string-greater-than" 和带有初始参数的函数 "urn:oasis:names:tc:xacml:1.0:function:string-equal" 两个参数的逻辑函数 "urn:oasis:names:tc:xacml:1.0:function:or" 进行赋值。

urn:oasis:names:tc:xacml:1.0:function:string-less-than

此函数将带有两个数据类型为 "http://www.w3.org/2001/XMLSchema#string" 的参数，并返回一个 "http://www.w3.org/2001/XMLSchema#boolean"。当且仅当下面这种情况，此函数返回 "True"，其参数逐字节进行比较，并且在来自经函数 "urn:oasis:names:tc:xacml:1.0:function:integer-equal" 计算认为相等的两个参数的相应字节的初始前缀之后，下一个逐字节的比较就是使用函数 "urn:oasis:names:tc:xacml:1.0:function:integer-less-than" 比较来自第一个参数的字节小于来自第二个参数的字节。否则它将返回 "False"。

urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal

此函数应带有两个数据类型为 "http://www.w3.org/2001/XMLSchema#string" 的参数，并应返回一个 "http://www.w3.org/2001/XMLSchema#boolean"。它应返回一个结果，就好像通过带有包含函数 "urn:oasis:names:tc:xacml:1.0:function:string-less-than" 和带有初始参数的函数 "urn:oasis:names:tc:xacml:1.0:function:string-equal" 两个参数的函数 "urn:oasis:names:tc:xacml:1.0:function:or" 进行赋值。

urn:oasis:names:tc:xacml:1.0:function:time-greater-than

此函数将带有两个数据类型为 "http://www.w3.org/2001/XMLSchema#time" 的参数，并应返回一个 "http://www.w3.org/2001/XMLSchema#boolean" 类型。当且仅当根据对 http://www.w3.org/2001/XMLSchema#time (W3C Signature:2002, 3.2.8) 指定的顺序关系，第一个参数大于第二个参数，此函数返回 "True"，否则它将返回 "False"。

注 1 — 不适合比较一个带有时区值的时间和不带时区值的时间。在这种情况下，应使用 time-in-range 函数。

```
urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal
```

此函数将带有两个为 "http://www.w3.org/2001/XMLSchema#time" 的参数，并应返回一个 "http://www.w3.org/2001/XMLSchema#boolean"。当且仅当根据对 "http://www.w3.org/2001/XMLSchema#time" (W3C Datatypes:2001, 3.2.8) 指定的顺序关系，第一个参数大于或等于第二个参数，此函数返回 "True"，否则它应返回 "False"。

注 2 — 不适合比较一个带有时区值的时间和不带时区值的时间。在这种情况下，应使用 time-in-range 函数。

```
urn:oasis:names:tc:xacml:1.0:function:time-less-than
```

此函数将带有两个为 "http://www.w3.org/2001/XMLSchema#time" 的参数，并应返回一个 "http://www.w3.org/2001/XMLSchema#boolean"。当且仅当根据对 http://www.w3.org/2001/XMLSchema#time (W3C Datatypes:2001, 3.2.8) 指定的顺序关系，第一个参数小于第二个参数，此函数返回 "True"，否则它应返回 "False"。

注 3 — 不适合比较一个带有时区值的时间和不带时区值的时间。在这种情况下，应使用 time-in-range 函数。

```
urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal
```

此函数将带有两个数据类型为 "http://www.w3.org/2001/XMLSchema#time" 的参数，并返回一个 "http://www.w3.org/2001/XMLSchema#boolean" 类型。当且仅当根据对 http://www.w3.org/2001/XMLSchema#time 指定的顺序关系，第一个参数小于或等于第二个参数，此函数返回 "True"，否则它将返回 "False"。

注 4 — 不适合比较一个带有时区值的时间和不带时区值的时间。在这种情况下，应使用 time-in-range 函数。

```
urn:oasis:names:tc:xacml:1.0:function:time-in-range
```

此函数将带有三个数据类型为 "http://www.w3.org/2001/XMLSchema#time" 的参数，并返回一个 "http://www.w3.org/2001/XMLSchema#boolean" 类型。当且仅当第一个参数在第二个和第三个参数定义的范围之内 (包括第二和第三个参数)，此函数返回 "True"，否则它将返回 "False"。不考虑它的值，第三个参数将被解释为一个等于第二个参数或比第二个参数晚少于 24 小时的时间。如果第一个参数不提供时区，它将在上下文的处理者中使用缺省时区。如果第二个参数和第三个参数不提供时区，那么它们将使用来自第一个参数的时区。

```
urn:oasis:names:tc:xacml:1.0:function:date-time-greater-than
```

此函数将带有两个数据类型为 "http://www.w3.org/2001/XMLSchema#dateTime" 的参数，并返回一个 "http://www.w3.org/2001/XMLSchema#boolean"。当且仅当根据由 W3C Datatype:2001, 3.2.7 节对 "http://www.w3.org/2001/XMLSchema#dateTime" 指定的顺序关系，第一个参数大于第二个参数，此函数返回 "True"，否则它将返回 "False"。

注 5 — 如果一个 dateTime 值中不包含时区值，则需要指定一个隐式的时区值，如 W3C Datatype:2001 中描述。

```
urn:oasis:names:tc:xacml:1.0:function:date-time-greater-than-or-equal
```

此函数将带有两个数据类型为 "http://www.w3.org/2001/XMLSchema#dateTime" 的参数，并返回一个 "http://www.w3.org/2001/XMLSchema#boolean" 类型。当且仅当根据由 W3C Datatype:2001, 3.2.7 节对 "http://www.w3.org/2001/XMLSchema#dateTime" 指定的顺序关系，第一个参数大于或等于第二个参数，此函数返回 "True"，否则它将返回 "False"。

注 6 — 如果一个 dateTime 值中不包含时区值，则需要指定一个隐式的时区值，如 W3C Datatype:2001 中描述。

```
urn:oasis:names:tc:xacml:1.0:function:date-time-less-than
```

此函数将带有两个数据类型为 "http://www.w3.org/2001/XMLSchema#dateTime" 的参数，并返回一个 "http://www.w3.org/2001/XMLSchema#boolean" 类型。当且仅当根据由 W3C Datatype:2001, 3.2.7 节对 "http://www.w3.org/2001/XMLSchema#dateTime" 指定的顺序关系，第一个参数小于第二个参数，此函数返回 "True"，否则它将返回 "False"。

注 7 — 如果一个dateTime值中不包含时区值，则需要指定一个隐式的时区值，如W3C Datatype:2001中描述。

```
urn:oasis:names:tc:xacml:1.0:function:date-time-less-than-or-equal
```

此函数将带有两个数据类型为"http://www.w3.org/2001/XMLSchema#dateTime"的参数，并返回一个"http://www.w3.org/2001/XMLSchema#boolean"类型。当且仅当根据由 W3C Datatype:2001, 3.2.7 节对"http://www.w3.org/2001/XMLSchema#dateTime"指定的顺序关系，第一个参数小于或等于第二个参数，此函数返回“True”，否则它将返回“False”。

注 8 — 如果一个dateTime值中不包含时区值，则需要指定一个隐式的时区值，如W3C Datatype:2001中描述。

```
urn:oasis:names:tc:xacml:1.0:function:date-greater-than
```

此函数将带有两个数据类型为"http://www.w3.org/2001/XMLSchema#date"的参数，并返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当根据对"http://www.w3.org/2001/XMLSchema#date"指定的顺序关系，第一个参数大于第二个参数，此函数返回“True”，否则它将返回“False”。

注 9 — 如果一个dateTime值中不包含时区值，则需要指定一个隐式的时区值，如W3C Datatype:2001中描述。

```
urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal
```

此函数将带有两个数据类型为"http://www.w3.org/2001/XMLSchema#date"的参数，并返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当根据对"http://www.w3.org/2001/XMLSchema#date"指定的顺序关系，第一个参数大于或等于第二个参数，此函数返回“True”，否则它将返回“False”。

注 10 — 如果一个dateTime值中不包含时区值，则需要指定一个隐式的时区值，如W3C Datatype:2001中描述。

```
urn:oasis:names:tc:xacml:1.0:function:date-less-than
```

此函数将带有两个数据类型为"http://www.w3.org/2001/XMLSchema#date"的参数，并返回一个"http://www.w3.org/2001/XMLSchema#boolean"。当且仅当根据对"http://www.w3.org/2001/XMLSchema#date"指定的顺序关系，第一个参数小于第二个参数，此函数返回“True”，否则它将返回“False”。

注 11 — 如果一个dateTime值中不包含时区值，则需要指定一个隐式的时区值，如W3C Datatype:2001中描述。

```
urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal
```

此函数将带有两个数据类型为"http://www.w3.org/2001/XMLSchema#date"的参数，并返回一个"http://www.w3.org/2001/XMLSchema#boolean"类型。当且仅当根据对"http://www.w3.org/2001/XMLSchema#date"指定的顺序关系，第一个参数小于或等于第二个参数，此函数返回“True”，否则它将返回“False”。

注 12 — 如果一个dateTime值中不包含时区值，则需要指定一个隐式的时区值，如W3C Datatype:2001中描述。

A.3.9 字符串函数

下面的函数对字符串和 URI 进行操作。

```
urn:oasis:names:tc:xacml:2.0:function:string-concatenate
```

此函数将带有两个或多个数据类型为"http://www.w3.org/2001/XMLSchema#string"的参数，并将返回"http://www.w3.org/2001/XMLSchema#string"。其结果将是参数的串联。

```
urn:oasis:names:tc:xacml:2.0:function:url-string-concatenate
```

此函数将带有一个数据类型为"http://www.w3.org/2001/XMLSchema#anyURI"的参数以及一个或多个数据类型为"http://www.w3.org/2001/XMLSchema#string"的参数，并返回一个"http://www.w3.org/2001/XMLSchema#anyURI"类型值。该函数的结果将是由“string”参数顺序添加到“anyURI”参数构造成的 URI。

A.3.10 包函数

这些函数对一包“类型”值进行操作，这里的“类型”是原语数据类型之一。为下面每个函数定义的附加条件将导致其赋值为“Indeterminate”的表达式。

```
urn:oasis:names:tc:xacml:1.0:function:type-one-and-only
```

该函数将带有一包“类型”值当作参数，并返回一个“类型”值。它将返回此包中的唯一值。如果此包中没有一个值，那么表达式结果将表示为“Indeterminate”。

```
urn:oasis:names:tc:xacml:1.0:function:type-bag-size
```

该函数将带有一包“type”值当作参数，并返回结果“<http://www.w3.org/2001/XMLSchema#integer>”来表示此包中值的数目。

```
urn:oasis:names:tc:xacml:1.0:function:type-is-in
```

该函数将带有“type”的参数作为第一参数，把一包 type 值作为第二参数，并将返回“<http://www.w3.org/2001/XMLSchema#boolean>”。当且仅当第一参数通过“urn:oasis:names:tc:xacml:x.x:function:type-equal”匹配此包中的任何值，该函数的结果将为“True”。否则它将返回“False”

```
urn:oasis:names:tc:xacml:1.0:function:type-bag
```

该函数将带有任何数目的“type”参数，并返回一包包含此参数值的“type”值。参数为 0 的此函数的应用将产生一个指定数据类型的空包。

A.3.11 集函数

这些函数通过从一个包中去除重复元素来作用于包的模仿集。

```
urn:oasis:names:tc:xacml:1.0:function:type-intersection
```

该函数将带有两个都是一包“类型”值的参数。它将返回一包“类型”值，这样它就仅包含那些两包中通用的元素，这个过程由“urn:oasis:names:tc:xacml:x.x:function:type-equal”决定。正如由“urn:oasis:names:tc:xacml:x.x:function:type-equal”决定的，结果中将没有重复。

```
urn:oasis:names:tc:xacml:1.0:function:type-at-least-one-member-of
```

该函数将带有两个都是一包“类型”值的参数。它将返回一个“<http://www.w3.org/2001/XMLSchema#boolean>”。当且仅当第二参数中包含至少一个第一参数中的元素时（如“urn:oasis:names:tc:xacml:x.x:function:type-is-in”决定的那样），此函数的结果为“True”。

```
urn:oasis:names:tc:xacml:1.0:function:type-union
```

该函数将带有两个都是一包“类型”值的参数，此表达式将返回一包“类型”，因此它包含两个包中的所有元素。正如在“urn:oasis:names:tc:xacml:x.x:function:type-equal”决定，所以结果中将没有重复。

```
urn:oasis:names:tc:xacml:1.0:function:type-subset
```

该函数将带有两个都是一包“类型”值的参数。它将返回一个“<http://www.w3.org/2001/XMLSchema#boolean>”。当且仅当第一参数是第二参数的一个子集时，该函数返回“True”。正因为由“urn:oasis:names:tc:xacml:x.x:function:type-equal”决定，在子集计算之前，每个参数都被当作已经在子集计算之前除去重复。

```
urn:oasis:names:tc:xacml:1.0:function:type-set-equals
```

该函数将带有两个都是一包“类型”值的参数，它将返回一个“<http://www.w3.org/2001/XMLSchema#boolean>”。它将返回对第一和第二参数的“urn:oasis:names:tc:xacml:x.x:function:type-subset”应用以及第二和第一参数的“urn:oasis:names:tc:xacml:x.x:function:type-subset”应用采用“urn:oasis:names:tc:xacml:1.0:function:and”的结果。

A.3.12 高阶包函数

本部分讨论了 XACML 中对包执行操作的那些函数，以便这些函数通常都可以应用到这些包中。

常规目的的函数语言可能对指定这些函数的语义是有利的（参见附录三，使用函数语言的一个信息性举例）

1) urn:oasis:names:tc:xacml:1.0:function:any-of

该函数在一个指定的原语值和一包值之间采用了一个布尔函数，当且仅当该包中至少一个元素的判断是“True”时，此函数将返回“True”。

该函数将带有三个参数。第一参数将是<xacml:Function>元素，该元素命名一个带有原语类型的两个参数的布尔函数。第二参数将是一个原语数据类型值。第三参数将是一包原语数据类型。这种表达式就好像在<xacml:Function>参数中命名的函数应用于第二参数，并且此包中的第三参数的每个元素和结果都与"urn:oasis:names:tc:xacml:1.0:function:or"相结合。

2) urn:oasis:names:tc:xacml:1.0:function:all-of

此函数在一个指定的原语值和一包值之间采用了一个布尔函数，当且仅当该包中每个元素的判断是“True”时，此函数将返回“True”。

该函数将带有三个参数，第一参数将是<xacml:Function>元素，该元素命名一个带有原语类型的两个参数的布尔函数。第二参数将是一个原语数据类型值。第三参数将是一包原语数据类型。这种表达式就好像在<xacml:Function>参数中命名的函数应用于第二参数，并且此包中的第三参数的每个元素和结果都用urn:oasis:names:tc:xacml:1.0:function:and组合。

3) urn:oasis:names:tc:xacml:1.0:function:any-of-any

此函数在一包值的每个元素和另一包值的每个元素之间采用了一个布尔函数，当且仅当该包中至少一个对比的判断是“True”时，此函数将返回“True”。

该函数将带有三个参数，第一参数将是<xacml:Function>元素，该元素命名一个带有原语类型的两个参数的布尔函数。第二参数将是一包原语数据类型。第三参数将是一包原语数据类型。这种表达式就好像在<xacml:Function>参数中命名的函数应用于第二参数和第三参数的每个元素之间，并且结果都使用urn:oasis:names:tc:xacml:1.0:function:or结合。该语义就是当且仅当来自这两包的元素的至少一个对比采用的判断是“True”时，该表达式的结果将是“True”。

4) urn:oasis:names:tc:xacml:1.0:function:all-of-any

该函数在两个包的元素之间采用了一个布尔函数。当且仅当第一个包的每个元素和第二个包的任意元素之间提供了判断是“True”时，该表达式将是“True”。

该函数将带有三个参数，第一参数将是<xacml:Function>元素，该元素命名一个带有原语类型的两个参数的布尔函数。第二参数将是一包原语数据类型。第三参数将是一包原语数据类型。这种表达式的结果就好像将“urn:oasis:names:tc:xacml:1.0:function:any-of”函数应用于第一个包的每个值，第二个包中的所有值使用提供的xacml:Function，并且结果将用“urn:oasis:names:tc:xacml:1.0:function:and”结合。

5) urn:oasis:names:tc:xacml:1.0:function:any-of-all

该函数在两个包的元素之间采用了一个布尔函数。当且仅当第二个包的每个元素和第一个包的任意元素之间提供的判断是“True”时，该表达式将是“True”。

该函数将带有三个参数，第一参数将是<xacml:Function>元素，该元素命名一个带有原语类型的两个参数的布尔函数。第二参数将是一包原语数据类型。第三参数将是一包原语数据类型。这种表达式的结果就好像将“urn:oasis:names:tc:xacml:1.0:function:any-of”函数应用于第二个包的每个值，第一个包中的所有值使用提供的xacml:Function，并且结果将用“urn:oasis:names:tc:xacml:1.0:function:and”结合。

6) urn:oasis:names:tc:xacml:1.0:function:all-of-all

该函数在两个包的元素之间采用了一个布尔函数。当且仅当第一个包的每个元素全部与第二个包的所有元素相反，而且其间提供的判断是“True”时，该表达式将是“True”。

该函数将带有三个参数，第一参数将是<xacml:Function>元素，该元素命名一个带有原语类型的两个参数的布尔函数。第二参数将是一包原语数据类型。第三参数将是一包原语数据类型。这种表达式的结果就好像将以<xacml:Function>元素命名的函数应用于第二个参数的每个元素和第三个参数的每个元素，并且结果将用“urn:oasis:names:tc:xacml:1.0:function:and”结合。这里的语义就是，当且仅当第一个包的所有元素相对于第二个包的所有元素采用了判断“True”时，该表达式的结果为“True”。

7) urn:oasis:names:tc:xacml:1.0:function:map

该函数将一包值转换为另一包值。

此函数将带有两个参数，第一个函数将是一个由<xacml:Function>元素命名的函数，该函数带有一个原语数据类型参数，并返回一个原语数据类型值。第二个参数将是一包原语数据类型。这种表达式的结果就好像将以<xacml:Function>元素命名的函数应用于此包中的每个元素，并导致整个包的值发生转换，其结果将是由以<xacml:Function>元素命名的函数返回的一包原语数据类型。

A.3.13 基于常规表达式的函数

这些函数对各种使用常规表达式的类型进行操作，并相当于“http://www.w3.org/2001/XMLSchema#boolean”。

```
urn:oasis:names:tc:xacml:1.0:function:string-regexp-match
```

该函数决定一个常规表达式匹配。它将带有两个“http://www.w3.org/2001/XMLSchema#string”参数并将返回“http://www.w3.org/2001/XMLSchema#boolean”。第一个参数将是一个常规表达式，第二个参数将是一个通用的字符串。当且仅当第二个参数与第一个参数的常规表达式样本的值匹配时，该函数将返回“True”。常规表达式的表达式将是在 W3C Datatypes:2001 的附录 F 中定义的那样，增加了下面这些样本表达式和规则：

- X?? 与 X 匹配不多于一次
- X*? 与 X 匹配任何次数，包括0
- X+? 与 X 匹配至少一次
- X{n}? 与 X 匹配n次
- X(n)? 与 X 匹配刚好n次
- X{n,m}? 与 X 匹配至少n次，但不多于m次

一旦使用这些附加的样本表达式之一，常规表达式将与第一个与该样本一致的参数的最小的子串匹配。

除了这些附加表达式外，常规表达式将与第一个与该样本一致的参数的最大子串匹配。

除了当分别使用字符“^”和“\$”准确的确定字符串的开始和结束，如果该样本二个参数的任意子串匹配，那么认为该样本是匹配的。

所有已证实的执行都将支持该指定表达式样本。可以支持附加的常规表达式样本。

```
urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match
```

该函数决定了一个常规表达式匹配。它将带有两个参数；第一个是类型“http://www.w3.org/2001/XMLSchema#string”，第二个是类型“http://www.w3.org/2001/XMLSchema#anyURI”。它将返回“http://www.w3.org/2001/XMLSchema#boolean”。第一个参数将是一个常规表达式，第二个参数是一个 URI。此函数将第二个参数转换为类型“http://www.w3.org/2001/XMLSchema#string”，然后应用“urn:oasis:names:tc:xacml:1.0:function:string-regexp-match”。

```
urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match
```

该函数决定了一个常规表达式匹配。它将带有两个参数；第一个是类型“http://www.w3.org/2001/XMLSchema#string”，第二个是类型“urn:oasis:names:tc:xacml:2.0:data-type:ipAddress”。它将返回“http://www.w3.org/2001/XMLSchema#boolean”。第一个参数是一个常规表达式，第二个参数是一个 IPv4 或 IPv6 地址。此函数将第二个参数转换为类型“http://www.w3.org/2001/XMLSchema#string”，然后应用“urn:oasis:names:tc:xacml:1.0:function:string-regexp-match”。

```
urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match
```

该函数决定了一个常规表达式匹配。它将带有两个参数；第一个是类型“http://www.w3.org/2001/XMLSchema#string”，第二个是类型“urn:oasis:names:tc:xacml:2.0:data-type:dnsName”。它将返回“http://www.w3.org/2001/XMLSchema#boolean”。第一个参数是一个常规表达式，第二个参数是一个 DNS 域名。此函数将第二个参数转换为类型“http://www.w3.org/2001/XMLSchema#string”，然后应用“urn:oasis:names:tc:xacml:1.0:function:string-regexp-match”。

```
urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match
```

该函数决定了一个常规表达式匹配。它将带有两个参数，第一个是类型“http://www.w3.org/2001/XMLSchema#string”，第二个是类型“urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name”。它将返回“http://www.w3.org/2001/XMLSchema#boolean”。第一个参数是一个常规表达式，第二个参数是一个 RFC 822 的名字。此函数将第二个参数转换为类型“http://www.w3.org/2001/XMLSchema#string”，然后应用“urn:oasis:names:tc:xacml:1.0:function:string-regexp-match”。

```
urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match
```

该函数决定了一个常规表达式匹配。它将带有两个参数；第一个是类型“http://www.w3.org/2001/XMLSchema#string”，第二个是类型“urn:oasis:names:tc:xacml:1.0:data-type:x500Name”。它将返回“http://www.w3.org/2001/XMLSchema#boolean”。第一个参数是一个常规表达式，第二个参数是一个 X.500 号码簿名。此函数将第二个参数转换为类型“http://www.w3.org/2001/XMLSchema#string”，然后应用“urn:oasis:names:tc:xacml:1.0:function:string-regexp-match”。

A.3.14 特殊匹配函数

这些函数对各种类型操作，根据特定的标准匹配算法，它们相当于“http://www.w3.org/2001/XMLSchema#boolean”。

```
urn:oasis:names:tc:xacml:1.0:function:x500Name-match
```

该函数将带有“urn:oasis:names:tc:xacml:2.0:data-type:x500Name”的两个参数，并返回“http://www.w3.org/2001/XMLSchema#boolean”。当且仅当使用 x500 的 Name-equal 比较第一个参数与来自第二个参数的 RDN 终端序列相匹配时，该函数返回“True”。

```
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match
```

该函数将带有两个参数，第一个是数据类型“http://www.w3.org/2001/XMLSchema#string”，第二个是数据类型“urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name”，并返回“http://www.w3.org/2001/XMLSchema#boolean”。根据 W3C Datatypes:2001, 3.2.1，当且仅当第一个参数与第二个参数匹配时此函数结果为“True”。

一个 IETF RFC822 的名字由本地部分，后随"@",后随域部分组成。本地部分对大小写敏感，而域部分（通常是一个 DNS 名）对大小写不敏感。

第二个参数包括一个完整的 rfc822 名。第一个参数是完整的 rfc822 名或其一部分，此 RFC822 名被用来在第二个参数中选择适当的值，如下所述。

为在第二个参数中匹配一个特定的地址，第一个参数必须指定完整的邮件地址以便匹配。为在第二个参数中匹配特定域的任意地址，第一个参数必须仅指定一个域名（通常是 DNS 名）。为在第二个参数中匹配特定域的任意地址，第一个参数指定设计的域部分的第一位必须用“.”。

A.3.15 基于XPath的函数

这部分规定了带有用 XPath 表达式的参数的函数。XPath 表达式相当于与该表达式匹配的一组 XML 节点的节点集，一个节点或节点集不在 XACML 的正式数据类型系统中。所有对节点集的比较或其他操作都独立于规定的特定函数集。也就是说，这些函数中的 XPath 表达式受限于 XACML 请求上下文。<xacml-context:Request> 元素是每个 XPath 表达式的上下文节点。定义了下列函数：

```
urn:oasis:names:tc:xacml:1.0:function:xpath-node-count
```

该函数带有参数“http://www.w3.org/2001/XMLSchema#string”，这将被解释为 XPath 表达式，而且其结果将是“http://www.w3.org/2001/XMLSchema#integer”。此函数的返回值是与给定 XPath 表达式匹配的节点集中节点的数量。

```
urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal
```

该函数带有两个“http://www.w3.org/2001/XMLSchema#string”参数，这将被解释为 XPath 表达式，而且返回“http://www.w3.org/2001/XMLSchema#boolean”。如果由第一个参数匹配的节点集中的任何 XML 节点等于由第二个参数匹配的节点集中的任何 XML 节点，那么此函数将返回“True”。具有相同身份的两个节点被认为是相等的。

```
urn:oasis:names:tc:xacml:1.0:function:xpath-node-match
```

该函数带有两个“http://www.w3.org/2001/XMLSchema#string”参数，这将被解释为 XPath 表达式，而且其结果将是“http://www.w3.org/2001/XMLSchema#boolean”。满足下面两个条件时，此函数的结果将是“True”。

- 1) 由第一个参数匹配的节点集中的任何 XML 节点与由第二个参数匹配的节点集中的任何 XML 节点是相等的；
- 2) 由第一个参数匹配的节点集中的任意 XML 节点下面的任何属性和元素节点等于由第二个参数匹配的节点集中的任何 XML 节点。

具有相同 ID 的两个节点被认为是相等的。

注 — 第一个条件相当于“xpath-node-equal”，并且保证“xpath-node-equal”是“xpath-node-match”的一个特例。

A.3.16 扩展函数和原语类型

由字符串标识符来规定函数和原语类型，这些字符串标识符考虑介绍那些除了由 XACML 规定的之外的函数。这一办法允许用特殊的函数和特定的原语数据类型来扩展 XACML 模块。

为保证 XACML 赋值策略的完整性，扩展函数的结果应仅依赖于它的参数值。全局性和隐藏的参数不应影响一个表达式的赋值。由于赋值顺序不能以一种标准的方式保证，函数不应有副作用。

附件 B

XACML标识符

本附件定义了常用实体的标准标识符。

B.1 XACML命名空间

目前，定义了两种 XACML 命名空间。

使用这个标识符来定义策略：

```
urn:oasis:names:tc:xacml:2.0:policy:schema:os
```

用该标识符来定义请求和响应上下文。

```
urn:oasis:names:tc:xacml:2.0:context:schema:os
```

B.2 接入主体类别

下面这个标识符指示发起接入请求的系统实体，也就是请求链中的初始实体。如果不特别规定主体类别，那这就是缺省值：

```
urn:oasis:names:tc:xacml:1.0:subject-category:access-subject
```

下面这个标识符表示将接收请求结果的系统实体（当它区别于接入主体时使用）。

```
urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject
```

该标识符表示传递接入请求的系统实体。可能会有多于 1 个这种标识符，它们传递该消息的顺序没有任何涵义。

```
urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject
```

该标识符表示与发起请求的本地或远端码库相关联的系统实体。相应的主体属性可能包括链接它的 URL 和码签名的特性。可能会有多于 1 个这种标识符。它们处理该请求的顺序没有任何涵义。

```
urn:oasis:names:tc:xacml:1.0:subject-category:codebase
```

该标识符表示与发起接入请求的计算机有关的系统实体。例如 IPSEC 实体。

```
urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine
```

B.3 数据类型

下面的标识符表示 A.2 中定义的数据类型。

```
urn:oasis:names:tc:xacml:2.0:data-types:dayTimeDuration  
urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration  
urn:oasis:names:tc:xacml:1.0:data-type:x500Name.  
urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name  
urn:oasis:names:tc:xacml:2.0:data-type:ipAddress  
urn:oasis:names:tc:xacml:2.0:data-type:dnsName
```

下面的数据类型标识符由 W3C DataTypes:2001 定义。

```
http://www.w3.org/2001/XMLSchema#string  
http://www.w3.org/2001/XMLSchema#boolean  
http://www.w3.org/2001/XMLSchema#integer  
http://www.w3.org/2001/XMLSchema#double  
http://www.w3.org/2001/XMLSchema#time  
http://www.w3.org/2001/XMLSchema#date  
http://www.w3.org/2001/XMLSchema#dateTime
```

```
http://www.w3.org/2001/XMLSchema#anyURI
http://www.w3.org/2001/XMLSchema#hexBinary
http://www.w3.org/2001/XMLSchema#base64Binary
```

B.4 主体属性

这些标识符指示一个主体的属性，使用时，它们将在请求上下文的<Subject>元素中出现。它们将通过<SubjectAttributeDesignator>元素或者指向请求上下文的<Subject>元素的<AttributeSelector>元素的方式接入。

这些属性中至少有一个与每个主体都有关联。每个与认证有关的包含在单个<Subject>元素中的属性都与同样的认证事件相关联。

下面这个标识符表示主体的名字。其缺省格式为“http://www.w3.org/2001/XMLSchema#string”。要引用其他格式，需使用附录 B.3 中列出的数据类型属性。

```
urn:oasis:names:tc:xacml:1.0:subject:subject-id
```

这些标识符指示主体的种类，缺省值为“access-subject”。

```
urn:oasis:names:tc:xacml:1.0:subject-category
```

这些标识符指示主体的安全域。它用来识别管理员和管理主体 id 所用名字空间的策略。

```
urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier
```

这些标识符指示用来证实主体身份的公共密钥。

```
urn:oasis:names:tc:xacml:1.0:subject:key-info
```

这些标识符指示主体认证的时间。

```
urn:oasis:names:tc:xacml:1.0:subject:authentication-time
```

这些标识符指示认证该主体所使用的方法。

```
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:authentication-method
```

根据 PEP，这些标识符指示该主体发起接入请求的时间。

```
urn:oasis:names:tc:xacml:1.0:subject:request-time
```

根据 PEP，这些标识符指示该主体当前会话开始的时间。

```
urn:oasis:names:tc:xacml:1.0:subject:session-start-time
```

该标识符指示认证证书激活的位置。它们确定为支持来自 SAML 认证状态的相应实体。

该标识符指示此位置用一个 IP 地址来表示。

```
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address
```

相应的属性应为数据类型“http://www.w3.org/2001/XMLSchema#string”。

该标识符指示此位置用一个 DNS 名字来表示。

```
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name
```

相应的属性应为数据类型“http://www.w3.org/2001/XMLSchema#string”。

由于已经在 LDAP 中定义了适当的属性，XACML 标识符应该通过在 IETF 对应 LDAP 的 RFC(参见 IETF RFC 2256) 的 URI 中增加属性名称形成。例如，在 IETF RFC 2256 中规定的 user password 的属性名应是：

```
http://www.ietf.org/rfc/rfc2256.txt#userPassword
```

B.5 资源属性

该标识符指示资源的属性。相应的属性可能会出现在请求上下文的<Resource>元素中，并且以<ResourceAttributeDesignator>元素或者指向请求上下文<Resource>元素的<AttributeSelector>元素的方式被访问。

该标识符标识请求接入的资源。如果提供了<xacml-context:ResourceContent>元素，那么请求接入的资源将是或者在<xacml-context:ResourceContent>元素中提供的所有的或部分资源。

```
urn:oasis:names:tc:xacml:1.0:resource:resource-id
```

下面这个属性标识<xacml-context:ResourceContent>元素内容顶级元素的名字空间。在请求上下文中提供的资源内容中规定资源名字空间的情况下，PDP 应该确认该属性定义的名字空间与资源中定义的是否一致。相应属性的类型应该是“http://www.w3.org/2001/XMLSchema#anyURI”。

```
urn:oasis:names:tc:xacml:2.0:resource:target-namespace
```

B.6 动作属性

该标识符指示被请求的属性的属性。使用时，它们将出现在请求上下文的<Action>元素中。并且应该以<ActionAttributeDesignator>元素或者指向请求上下文<Action>元素的<AttributeSelector>元素的方式被接入。

该属性表示请求接入的动作。

```
urn:oasis:names:tc:xacml:1.0:action:action-id
```

由于动作是隐藏的，动作 ID 的属性值应该是：

```
urn:oasis:names:tc:xacml:1.0:action:implied-action
```

该属性指示定义动作 ID 属性的名字空间。

```
urn:oasis:names:tc:xacml:1.0:action:action-namespace
```

B.7 环境属性

这些标识符指示赋值判决请求所处的环境属性。在判决请求中使用，它们将出现在该请求上下文的<Environment>元素中。它们应该以<EnvironmentAttributeDesignator>元素或者指向请求上下文<Environment>元素的<AttributeSelector>元素的方式被访问。

下面这个标识符表示上下文处理器的当前时间。在实际应用中，它就是创建请求上下文的时间。为此，如果这些标识符出现在<Policy>或<PolicySet>中的多个地方，那么该赋值过程中的每个事件都应分配同样的值，不管处理这些事件期间消耗了多少时间。

```
urn:oasis:names:tc:xacml:1.0:environment:current-time
```

相应的属性应该是数据类型“http://www.w3.org/2001/XMLSchema#time”。

```
urn:oasis:names:tc:xacml:1.0:environment:current-date
```

相应的属性应该是数据类型“http://www.w3.org/2001/XMLSchema#date”。

```
urn:oasis:names:tc:xacml:1.0:environment:current-dateTime
```

相应的属性应该是数据类型“http://www.w3.org/2001/XMLSchema#dateTime”。

B.8 状态码

这里定义了下列状态码的值。

该标识符指示成功。

```
urn:oasis:names:tc:xacml:1.0:status:ok
```

该标识符指示制定策略决定的所有必要属性都不可用。

```
urn:oasis:names:tc:xacml:1.0:status:missing-attribute
```

该标识符指示语法错误中所包含的一些属性值，例如，在数字域里出现字母等语法错误。

```
urn:oasis:names:tc:xacml:1.0:status:syntax-error
```

该标识符指示在赋值策略的过程中出现一个错误，由 0 分割开一个例子。

```
urn:oasis:names:tc:xacml:1.0:status:processing-error
```

B.9 组合算法

deny-overrides 规则组合算法，对 ruleCombiningAlgId 属性值如下：

```
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides
```

deny-overrides 策略组合算法，对 policyCombiningAlgId 属性值如下：

```
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides
```

Permit-overrides 规则组合算法，对 ruleCombiningAlgId 属性值如下：

```
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:Permit-overrides
```

Permit-overrides 策略组合算法，对 policyCombiningAlgId 属性值如下：

```
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:Permit-overrides
```

first-applicable 规则组合算法，对 ruleCombiningAlgId 属性值如下：

```
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable
```

first-applicable 策略组合算法，对 policyCombiningAlgId 属性值如下：

```
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable
```

only-one-applicable-policy 策略组合算法，对 policyCombiningAlgId 属性值如下：

```
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable
```

ordered-deny-overrides 规则组合算法，对 ruleCombiningAlgId 属性值如下：

```
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides
```

ordered-deny-overrides 策略组合算法，对 policyCombiningAlgId 属性值如下：

```
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides
```

ordered-Permit-overrides 规则组合算法，对 ruleCombiningAlgId 属性值如下：

```
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-Permit-overrides
```

ordered-Permit-overrides 策略组合算法，对 policyCombiningAlgId 属性值如下：

```
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-Permit-overrides
```

附件 C

组合算法

本附件包含由 XACML 所确定的规则和策略组合算法的描述。

C.1 Deny-overrides

C.1.1 本节定义了一个策略的“Deny-overrides”规则组合算法。

在该策略的整套规则中，一旦任何规则的赋值为“Deny”，该规则组合的结果将为“Deny”。如果任何规则的赋值为“Permit”并且所有其他规则的赋值为“NotApplicable”，那么该规则组合的结果将是“Permit”。换句话说，就是“Deny”值优先，不考虑该组合中其他规则的赋值结果。如果发现判决请求的所有规则都是“NotApplicable”，那么该规则组合的赋值为“NotApplicable”。

如果在赋值目标或一个包含效果值“Deny”规则的过程中出现一个错误，赋值过程将为寻找赋值“Deny”而继续赋值后续规则。如果没有其他规则的赋值为“Deny”，那么该规则组合将被赋值为“Indeterminate”，并带有相应的错误状态。

如果至少一个规则的赋值为“Permit”，所有没有出现赋值错误的其他规则的赋值为“Permit”或“NotApplicable”，并且所有出现赋值错误的规则包含效果“Permit”，那么该规则组合的赋值结果将为“Permit”。

下面的伪代码代表这个规则组合算法的赋值策略。

```
Decision denyOverridesRuleCombiningAlgorithm(Rule rule[])
{
    Boolean atLeastOneError = false;
    Boolean potentialDeny = false;
    Boolean atLeastOnePermit = false;
    for( i=0 ; i < lengthOf(rules) ; i++ )
    {
        Decision decision = evaluate(rule[i]);
        if (decision == Deny)
        {
            return Deny;
        }
        if (decision == Permit)
        {
            atLeastOnePermit = true;
            continue;
        }
        if (decision == NotApplicable)
        {
            continue;
        }
        if (decision == Indeterminate)
        {
            atLeastOneError = true;

            if (effect(rule[i]) == Deny)
            {
                potentialDeny = true;
            }
            continue;
        }
    }
    if (potentialDeny)
    {
        return Indeterminate;
    }
    if (atLeastOnePermit)
    {
        return Permit;
    }
    if (atLeastOneError)
    {
```

```

        return Indeterminate;
    }
    return NotApplicable;
}

```

C.1.2 本部分定义了一个策略集的“Deny-overrides”策略组合算法。

在该策略集的所有策略集中，如果任何策略的赋值为“Deny”，那么该策略组合的结果将为“Deny”。换句话说，“Deny”值优先，不考虑该策略集中其他策略的赋值结果。如果发现判决请求的所有策略都是“NotApplicable”，那么该策略集的赋值为“NotApplicable”。

如果在赋值一个策略的目标的或者参考一个策略被认为是无效或者该策略赋值为“Indeterminate”，那么该策略集将被赋值为“Deny”。

下面的伪代码代表这个策略组合算法的赋值策略。

```

Decision denyOverridesPolicyCombiningAlgorithm(Policy policy[])
{
    Boolean atLeastOnePermit = false;
    for( i=0 ; i < lengthOf(policy) ; i++ )
    {
        Decision decision = evaluate(policy[i]);
        if (decision == Deny)
        {
            return Deny;
        }
        if (decision == Permit)
        {
            atLeastOnePermit = true;
            continue;
        }
        if (decision == NotApplicable)
        {
            continue;
        }
        if (decision == Indeterminate)
        {
            return Deny;
        }
    }
    if (atLeastOnePermit)
    {
        return Permit;
    }
    return NotApplicable;
}

```

单个策略的职责将被结合进来，参见 7.6.14 节所述。

C.2 Ordered-deny-overrides

下面的段落定义了一个策略的“Ordered-deny-overrides”规则组合算法。

除了一个例外，该算法的动作与 Deny-overrides 规则组合算法的动作相同。所收集规则的赋值顺序将与该策略中列出的顺序匹配。

下面的段落定义了一个策略集的“Ordered-deny-overrides”策略组合算法。

除了一个例外，该算法的动作与 Deny-overrides 策略组合算法的动作相同。所收集策略的赋值顺序将与该策略集中列出的顺序匹配。

C.3 Permit-overrides

C.3.1 本部分定义了一个策略的“Permit-overrides”规则组合算法。

在该策略的整套规则中，一旦任何规则的赋值为“Permit”，该规则组合的结果将为“Permit”。如果任何规则的赋值为“Deny”并且所有其他规则的赋值为“NotApplicable”，那么该规则组合的结果将是“Deny”。换句话说，就是“Permit”值优先，不考虑该组合中其他规则的赋值结果。如果发现判决请求的所有规则都是“NotApplicable”，那么该规则组合的赋值为“NotApplicable”。

如果在赋值目标或一个包含效果值“Permit”规则的过程中出现一个错误，赋值过程将继续寻找赋值为“Permit”。如果没有其他规则的赋值为“Permit”，那么该策略将被赋值为“Indeterminate”，并带有相应的错误状态。

如果至少一个规则的赋值为“Deny”，所有没有出现赋值错误的其他规则的赋值为“Deny”或“NotApplicable”，并且所有出现赋值错误的规则包含效果值“Deny”，那么该策略的赋值结果将为“Deny”。

下面的伪代码代表这个规则组合算法的赋值策略。

```
Decision permitOverridesRuleCombiningAlgorithm(Rule rule[])
{
    Boolean atLeastOneError = false;
    Boolean potentialPermit = false;
    Boolean atLeastOneDeny = false;
    for( i=0 ; i < lengthOf(rule) ; i++ )
    {
        Decision decision = evaluate(rule[i]);
        if (decision == Deny)
        {
            atLeastOneDeny = true;
            continue;
        }
        if (decision == Permit)
        {
            return Permit;
        }
        if (decision == NotApplicable)
        {
            continue;
        }
        if (decision == Indeterminate)
        {
            atLeastOneError = true;

            if (effect(rule[i]) == Permit)
            {
                potentialPermit = true;
            }
            continue;
        }
    }
    if (potentialPermit)
    {
        return Indeterminate;
    }
    if (atLeastOneDeny)
    {
        return Deny;
    }
    if (atLeastOneError)
    {
        return Indeterminate;
    }
    return NotApplicable;
}
```

C.3.2 本部分定义了一个策略集的“Permit-overrides”策略组合算法

在该策略集的所有策略中，如果任何策略的赋值为“Permit”，那么该策略组合的结果将为“Permit”。换句话说，“Permit”值优先，不考虑该策略集中其他策略的赋值结果。如果发现判决请求的所有策略都是“NotApplicable”，那么该策略集的赋值将是“NotApplicable”。

如果在赋值一个策略的目标的过程中或者一个策略的参考被认为是非法或者该策略赋值为“Indeterminate”时出错，假若没有其他策略被赋值为“Permit”或“Deny”，那么该策略集将被赋值为“Indeterminate”，并带有适当的错误状态。

下面的伪代码代表对这个策略组合算法的赋值策略。

```
Decision permitOverridesPolicyCombiningAlgorithm(Policy policy[])
{
    Boolean atLeastOneError = false;
    Boolean atLeastOneDeny = false;
    for( i=0 ; i < lengthOf(policy) ; i++ )
    {
        Decision decision = evaluate(policy[i]);
        if (decision == Deny)
        {
            atLeastOneDeny = true;
            continue;
        }
        if (decision == Permit)
        {
            return Permit;
        }
        if (decision == NotApplicable)
        {
            continue;
        }
        if (decision == Indeterminate)
        {
            atLeastOneError = true;
            continue;
        }
    }
    if (atLeastOneDeny)
    {
        return Deny;
    }
    if (atLeastOneError)
    {
        return Indeterminate;
    }
    return NotApplicable;
}
```

单个策略的职责将被结合进来，参见 7.6.14 节所述。

C.4 Ordered-Permit-overrides

下面的段落定义了一个策略的“Ordered-Permit-overrides”规则组合算法。

除了一个例外，该算法的行为与 Permit-overrides 规则组合算法的行为相同。所收集规则的赋值顺序将与该策略中列出的顺序匹配。

下面的段落定义了一个策略集的“Ordered-Permit-overrides”策略组合算法。

除了一个例外，该算法的行为与 Permit-overrides 策略组合算法的行为相同。所收集策略的赋值顺序将与该策略集中列出的顺序匹配。

C.5 First-applicable

C.5.1 本部分定义了一个策略的“First-Applicable”规则组合算法

每个规则将按照在策略中列出的顺序进行赋值。对于一个特定规则，如果目标匹配并且条件赋值为“True”，那么该策略的赋值将停止并且该规则的相应效果将作为策略的赋值结果（如：“Permit”或“Deny”）。对于赋值中选出的一个特定规则，如果目标赋值为“False”或条件赋值为“False”，那么按顺序的下一个规则将被赋值。如果按顺序没有后续规则的话，该策略将被赋值为“NotApplicable”。

如果在赋值一个规则的目标或条件的过程中出现一个错误，那么赋值过程将停止，并且该策略将被赋值为“Indeterminate”，并带有相应的错误状态。

下面的伪代码代表了规则组合算法的赋值策略。

```
Decision firstApplicableEffectRuleCombiningAlgorithm(Rule rule[])
{
    for( i = 0 ; i < lengthOf(rule) ; i++ )
    {
        Decision decision = evaluate(rule[i]);
        if (decision == Deny)
        {
            return Deny;
        }
        if (decision == Permit)
        {
            return Permit;
        }
        if (decision == NotApplicable)
        {
            continue;
        }
        if (decision == Indeterminate)
        {
            return Indeterminate;
        }
    }
    return NotApplicable;
}
```

C.5.2 本部分定义了一个策略集的“First-applicable”策略组合算法

每个策略将按照在策略集中列出的顺序进行赋值。对于一个特定策略，如果目标赋值为“True”并且该策略赋值为确定的值“Permit”或“Deny”，那么赋值将停止并且该策略的相应效果将作为策略集的赋值结果。对于一个特定策略，如果目标赋值为“False”或该策略赋值为“不可应用的”，那么按顺序的下一个策略将被赋值。如果按顺序没有后续策略的话，该策略集将被赋值为“不可应用的”。

如果在赋值目标的过程中出现一个错误，或者当赋值一个特定策略，即该策略的参考被认为是非法，或者该策略本身赋值为“Indeterminate”时，该策略组合算法的赋值过程将停止，并且该策略集将被赋值为“Indeterminate”，并带有相应的错误状态。

下面的伪代码代表了策略组合算法的赋值策略。

```
Decision firstApplicableEffectPolicyCombiningAlgorithm(Policy policy[])
{
    for( i = 0 ; i < lengthOf(policy) ; i++ )
    {
        Decision decision = evaluate(policy[i]);
        if(decision == Deny)
        {
            return Deny;
        }
        if(decision == Permit)
        {
            return Permit;
        }
        if (decision == NotApplicable)
        {
            continue;
        }
        if (decision == Indeterminate)
        {
            return Indeterminate;
        }
    }
    return NotApplicable;
}
```

单个策略的职责将被结合进来，参见 7.6.14 节所述。

C.6 Only-one-applicable

本部分定义了一个策略集的“Only-one-applicable”策略组合算法。

在该策略集的所有策略中，如果没有一个策略对其目标可用，那么该策略组合算法的结果将是“NotApplicable”。如果有多个策略对其目标可用，那么该策略组合算法的结果将是“Indeterminate”。

如果赋值目标时仅有一个策略可用，那么该策略组合算法的结果将作为该策略的赋值结果。

如果在赋值一个策略的目标时出现错误，或者一个策略的参考被认为非法，或者该策略赋值为“Indeterminate”，那么该策略集将赋值为“Indeterminate”，并带有相应的错误状态。

下面的伪代码代表了 this 策略组合算法的赋值策略。

```
Decision onlyOneApplicablePolicyPolicyCombiningAlogrithm(Policy policy[])
{
    Boolean          atLeastOne      = false;
    Policy           selectedPolicy = null;
    ApplicableResult appResult;

    for ( i = 0; i < lengthOf(policy) ; i++ )
    {
        appResult = isApplicable(policy[i]);

        if ( appResult == Indeterminate )
        {
            return Indeterminate;
        }
        if( appResult == Applicable )
        {
            if ( atLeastOne )
            {
                return Indeterminate;
            }
            else
            {
                atLeastOne      = true;
                selectedPolicy = policy[i];
            }
        }
        if ( appResult == NotApplicable )
        {
            continue;
        }
    }
    if ( atLeastOne )
    {
        return evaluate(selectedPolicy);
    }
    else
    {
        return NotApplicable;
    }
}
```

附件 D

XACML模式定义

D.1 XACML 上下文模式定义

本部分提供 XACML 上下文模式定义。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:import namespace="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    schemaLocation="http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-
os.xsd"/>
  <!-- -->
  <xs:element name="Request" type="xacml-context:RequestType"/>
  <xs:complexType name="RequestType">
    <xs:sequence>
      <xs:element ref="xacml-context:Subject" maxOccurs="unbounded"/>
      <xs:element ref="xacml-context:Resource" maxOccurs="unbounded"/>
      <xs:element ref="xacml-context:Action"/>
      <xs:element ref="xacml-context:Environment"/>
    </xs:sequence>
  </xs:complexType>
  <!-- -->
  <xs:element name="Response" type="xacml-context:ResponseType"/>
  <xs:complexType name="ResponseType">
    <xs:sequence>
      <xs:element ref="xacml-context:Result" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- -->
  <xs:element name="Subject" type="xacml-context:SubjectType"/>
  <xs:complexType name="SubjectType">
    <xs:sequence>
      <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="SubjectCategory" type="xs:anyURI"
    default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
  </xs:complexType>
  <!-- -->
  <xs:element name="Resource" type="xacml-context:ResourceType"/>
  <xs:complexType name="ResourceType">
    <xs:sequence>
      <xs:element ref="xacml-context:ResourceContent" minOccurs="0"/>
      <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <!-- -->
  <xs:element name="ResourceContent" type="xacml-context:ResourceContentType"/>
  <xs:complexType name="ResourceContentType" mixed="true">
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>
  <!-- -->
  <xs:element name="Action" type="xacml-context:ActionType"/>
  <xs:complexType name="ActionType">
    <xs:sequence>
      <xs:element ref="xacml-context:Attribute" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

```



```

</xs:complexType>
<!-- -->
<xs:element name="Environment" type="xacml-context:EnvironmentType"/>
<xs:complexType name="EnvironmentType">
  <xs:sequence>
    <xs:element ref="xacml-context:Attribute" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Attribute" type="xacml-context:AttributeType"/>
<xs:complexType name="AttributeType">
  <xs:sequence>
    <xs:element ref="xacml-context:AttributeValue" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
  <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
  <xs:attribute name="Issuer" type="xs:string" use="optional"/>
</xs:complexType>
<!-- -->
<xs:element name="AttributeValue" type="xacml-context:AttributeValueType"/>
<xs:complexType name="AttributeValueType" mixed="true">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>
<!-- -->
<xs:element name="Result" type="xacml-context:ResultType"/>
<xs:complexType name="ResultType">
  <xs:sequence>
    <xs:element ref="xacml-context:Decision"/>
    <xs:element ref="xacml-context:Status" minOccurs="0"/>
    <xs:element ref="xacml:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="ResourceId" type="xs:string" use="optional"/>
</xs:complexType>
<!-- -->
<xs:element name="Decision" type="xacml-context:DecisionType"/>
<xs:simpleType name="DecisionType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Permit"/>
    <xs:enumeration value="Deny"/>
    <xs:enumeration value="Indeterminate"/>
    <xs:enumeration value="NotApplicable"/>
  </xs:restriction>
</xs:simpleType>
<!-- -->
<xs:element name="Status" type="xacml-context:StatusType"/>
<xs:complexType name="StatusType">
  <xs:sequence>
    <xs:element ref="xacml-context:StatusCode"/>
    <xs:element ref="xacml-context:StatusMessage" minOccurs="0"/>
    <xs:element ref="xacml-context:StatusDetail" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="StatusCode" type="xacml-context:StatusCodeType"/>
<xs:complexType name="StatusCodeType">
  <xs:sequence>
    <xs:element ref="xacml-context:StatusCode" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="Value" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="StatusMessage" type="xs:string"/>
<!-- -->
<xs:element name="StatusDetail" type="xacml-context:StatusDetailType"/>
<xs:complexType name="StatusDetailType">

```

```

        <xs:sequence>
            <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <!-- -->
    <xs:element name="MissingAttributeDetail" type="xacml-
context:MissingAttributeDetailType"/>
    <xs:complexType name="MissingAttributeDetailType">
        <xs:sequence>
            <xs:element ref="xacml-context:AttributeValue" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
        <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
        <xs:attribute name="Issuer" type="xs:string" use="optional"/>
    </xs:complexType>
    <!-- -->
</xs:schema>

```

D.2 策略模式定义

本部分提供 XACML 策略模式定义。

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <!-- -->
    <xs:element name="PolicySet" type="xacml:PolicySetType"/>
    <xs:complexType name="PolicySetType">
        <xs:sequence>
            <xs:element ref="xacml:Description" minOccurs="0"/>
            <xs:element ref="xacml:PolicySetDefaults" minOccurs="0"/>
            <xs:element ref="xacml:Target"/>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element ref="xacml:PolicySet"/>
                <xs:element ref="xacml:Policy"/>
                <xs:element ref="xacml:PolicySetIdReference"/>
                <xs:element ref="xacml:PolicyIdReference"/>
                <xs:element ref="xacml:CombinerParameters"/>
                <xs:element ref="xacml:PolicyCombinerParameters"/>
                <xs:element ref="xacml:PolicySetCombinerParameters"/>
            </xs:choice>
            <xs:element ref="xacml:Obligations" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
        <xs:attribute name="Version" type="xacml:VersionType" default="1.0"/>
        <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
    </xs:complexType>
    <!-- -->
    <xs:element name="CombinerParameters" type="xacml:CombinerParametersType"/>
    <xs:complexType name="CombinerParametersType">
        <xs:sequence>
            <xs:element ref="xacml:CombinerParameter" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <!-- -->
    <xs:element name="CombinerParameter" type="xacml:CombinerParameterType"/>
    <xs:complexType name="CombinerParameterType">
        <xs:sequence>
            <xs:element ref="xacml:AttributeValue"/>
        </xs:sequence>
        <xs:attribute name="ParameterName" type="xs:string" use="required"/>
    </xs:complexType>
    <!-- -->
    <xs:element name="RuleCombinerParameters"
type="xacml:RuleCombinerParametersType"/>

```

```

<xs:complexType name="RuleCombinerParametersType">
  <xs:complexContent>
    <xs:extension base="xacml:CombinerParametersType">
      <xs:attribute name="RuleIdRef" type="xs:string"
use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="PolicyCombinerParameters"
type="xacml:PolicyCombinerParametersType"/>
<xs:complexType name="PolicyCombinerParametersType">
  <xs:complexContent>
    <xs:extension base="xacml:CombinerParametersType">
      <xs:attribute name="PolicyIdRef" type="xs:anyURI"
use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="PolicySetCombinerParameters"
type="xacml:PolicySetCombinerParametersType"/>
<xs:complexType name="PolicySetCombinerParametersType">
  <xs:complexContent>
    <xs:extension base="xacml:CombinerParametersType">
      <xs:attribute name="PolicySetIdRef" type="xs:anyURI"
use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="PolicySetIdReference" type="xacml:IdReferenceType"/>
<xs:element name="PolicyIdReference" type="xacml:IdReferenceType"/>
<!-- -->
<xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
<xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
<xs:complexType name="DefaultsType">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="xacml:XPathVersion"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="XPathVersion" type="xs:anyURI"/>
<!-- -->
<xs:complexType name="IdReferenceType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="Version" type="xacml:VersionMatchType"
use="optional"/>
      <xs:attribute name="EarliestVersion"
type="xacml:VersionMatchType" use="optional"/>
      <xs:attribute name="LatestVersion"
type="xacml:VersionMatchType" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<!-- -->
<xs:simpleType name="VersionType">
  <xs:restriction base="xs:string">
    <xs:pattern value="(\d+\.)*\d+"/>
  </xs:restriction>
</xs:simpleType>
<!-- -->
<xs:simpleType name="VersionMatchType">
  <xs:restriction base="xs:string">
    <xs:pattern value="((\d+|\*)\.)*(\d+|\*|\+)/>
  </xs:restriction>
</xs:simpleType>
<!-- -->

```

```

<xs:element name="Policy" type="xacml:PolicyType"/>
<xs:complexType name="PolicyType">
  <xs:sequence>
    <xs:element ref="xacml:Description" minOccurs="0"/>
    <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
    <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>
    <xs:element ref="xacml:Target"/>
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>
      <xs:element ref="xacml:RuleCombinerParameters" minOccurs="0"/>
      <xs:element ref="xacml:VariableDefinition"/>
      <xs:element ref="xacml:Rule"/>
    </xs:choice>
    <xs:element ref="xacml:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
  <xs:attribute name="Version" type="xacml:VersionType" default="1.0"/>
  <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="Description" type="xs:string"/>
<!-- -->
<xs:element name="Rule" type="xacml:RuleType"/>
<xs:complexType name="RuleType">
  <xs:sequence>
    <xs:element ref="xacml:Description" minOccurs="0"/>
    <xs:element ref="xacml:Target" minOccurs="0"/>
    <xs:element ref="xacml:Condition" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="RuleId" type="xs:string" use="required"/>
  <xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
</xs:complexType>
<!-- -->
<xs:simpleType name="EffectType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Permit"/>
    <xs:enumeration value="Deny"/>
  </xs:restriction>
</xs:simpleType>
<!-- -->
<xs:element name="Target" type="xacml:TargetType"/>
<xs:complexType name="TargetType">
  <xs:sequence>
    <xs:element ref="xacml:Subjects" minOccurs="0"/>
    <xs:element ref="xacml:Resources" minOccurs="0"/>
    <xs:element ref="xacml:Actions" minOccurs="0"/>
    <xs:element ref="xacml:Environments" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Subjects" type="xacml:SubjectsType"/>
<xs:complexType name="SubjectsType">
  <xs:sequence>
    <xs:element ref="xacml:Subject" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Subject" type="xacml:SubjectType"/>
<xs:complexType name="SubjectType">
  <xs:sequence>
    <xs:element ref="xacml:SubjectMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Resources" type="xacml:ResourcesType"/>
<xs:complexType name="ResourcesType">
  <xs:sequence>
    <xs:element ref="xacml:Resource" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->

```

```

<xs:element name="Resource" type="xacml:ResourceType"/>
<xs:complexType name="ResourceType">
  <xs:sequence>
    <xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Actions" type="xacml:ActionTypes"/>
<xs:complexType name="ActionTypes">
  <xs:sequence>
    <xs:element ref="xacml:Action" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Action" type="xacml:ActionType"/>
<xs:complexType name="ActionType">
  <xs:sequence>
    <xs:element ref="xacml:ActionMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Environments" type="xacml:EnvironmentsType"/>
<xs:complexType name="EnvironmentsType">
  <xs:sequence>
    <xs:element ref="xacml:Environment" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Environment" type="xacml:EnvironmentType"/>
<xs:complexType name="EnvironmentType">
  <xs:sequence>
    <xs:element ref="xacml:EnvironmentMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="SubjectMatch" type="xacml:SubjectMatchType"/>
<xs:complexType name="SubjectMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:SubjectAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="ResourceMatch" type="xacml:ResourceMatchType"/>
<xs:complexType name="ResourceMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:ResourceAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="ActionMatch" type="xacml:ActionMatchType"/>
<xs:complexType name="ActionMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:ActionAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->

```

```

<xs:element name="EnvironmentMatch" type="xacml:EnvironmentMatchType"/>
<xs:complexType name="EnvironmentMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:EnvironmentAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="VariableDefinition" type="xacml:VariableDefinitionType"/>
<xs:complexType name="VariableDefinitionType">
  <xs:sequence>
    <xs:element ref="xacml:Expression"/>
  </xs:sequence>
  <xs:attribute name="VariableId" type="xs:string" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="Expression" type="xacml:ExpressionType" abstract="true"/>
<xs:complexType name="ExpressionType" abstract="true"/>
<!-- -->
<xs:element name="VariableReference"
type="xacml:VariableReferenceType" substitutionGroup="xacml:Expression"/>
<xs:complexType name="VariableReferenceType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="VariableId" type="xs:string"
use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="AttributeSelector"
type="xacml:AttributeSelectorType" substitutionGroup="xacml:Expression"/>
<xs:complexType name="AttributeSelectorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="RequestContextPath" type="xs:string"
use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI"
use="required"/>
      <xs:attribute name="MustBePresent" type="xs:boolean"
use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="ResourceAttributeDesignator"
type="xacml:AttributeDesignatorType" substitutionGroup="xacml:Expression"/>
<xs:element name="ActionAttributeDesignator"
type="xacml:AttributeDesignatorType" substitutionGroup="xacml:Expression"/>
<xs:element name="EnvironmentAttributeDesignator"
type="xacml:AttributeDesignatorType" substitutionGroup="xacml:Expression"/>
<!-- -->
<xs:complexType name="AttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="AttributeId" type="xs:anyURI"
use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI"
use="required"/>
      <xs:attribute name="Issuer" type="xs:string" use="optional"/>
      <xs:attribute name="MustBePresent" type="xs:boolean"
use="optional" default="false"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="SubjectAttributeDesignator"

```

```

type="xacml:SubjectAttributeDesignatorType" substitutionGroup="xacml:Expression"/>
<xs:complexType name="SubjectAttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:AttributeDesignatorType">
      <xs:attribute name="SubjectCategory" type="xs:anyURI"
        use="optional" default="urn:oasis:names:tc:xacml:1.0:subject-
category: access-subject"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="AttributeValue" type="xacml:AttributeValueType"
  substitutionGroup="xacml:Expression"/>
<xs:complexType name="AttributeValueType" mixed="true">
  <xs:complexContent mixed="true">
    <xs:extension base="xacml:ExpressionType">
      <xs:sequence>
        <xs:any namespace="##any" processContents="lax"
minOccurs="0"
          maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="DataType" type="xs:anyURI"
use="required"/>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="Function" type="xacml:FunctionType"
substitutionGroup="xacml:Expression"/>
<xs:complexType name="FunctionType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="Condition" type="xacml:ConditionType"/>
<xs:complexType name="ConditionType">
  <xs:sequence>
    <xs:element ref="xacml:Expression"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Apply" type="xacml:ApplyType"
substitutionGroup="xacml:Expression"/>
<xs:complexType name="ApplyType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:sequence>
        <xs:element ref="xacml:Expression" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="FunctionId" type="xs:anyURI"
use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="Obligations" type="xacml:ObligationsType"/>
<xs:complexType name="ObligationsType">
  <xs:sequence>
    <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<!-- -->
<xs:element name="Obligation" type="xacml:ObligationType"/>
<xs:complexType name="ObligationType">
  <xs:sequence>

```

```

        <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
    <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
</xs:complexType>
<!-- -->
<xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
<xs:complexType name="AttributeAssignmentType" mixed="true">
    <xs:complexContent mixed="true">
        <xs:extension base="xacml:AttributeValueType">
            <xs:attribute name="AttributeId" type="xs:anyURI"
use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<!-- -->
</xs:schema>

```

D.3 XACML SAML 协议模式定义

本部分提供 XACML SAML 协议模式定义。

```

<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="urn:oasis:xacml:2.0:saml:protocol:schema:os"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
    xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="http://www.oasis-
open.org/committees/tc_home.php?wg_abbrev=security"/>
    <xs:import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
        schemaLocation="http://www.oasis-
open.org/committees/tc_home.php?wg_abbrev=security"/>
    <xs:import namespace="urn:oasis:names:tc:xacml:2.0:context:schema:os"
        schemaLocation="http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-
context-schema-os.xsd"/>
    <xs:import namespace="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
        schemaLocation="http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-
policy-schema-os.xsd"/>
    <xs:annotation>
        <xs:documentation>
            Document identifier: access_control-xacml-2.0-saml-protocol-schema-os.xsd
            Location: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-
protocol-schema-os.xsd
        </xs:documentation>
    </xs:annotation>
    <!-- -->
    <xs:element name="XACMLAuthzDecisionQuery"
        type="XACMLAuthzDecisionQueryType"/>
    <xs:complexType name="XACMLAuthzDecisionQueryType">
        <xs:complexContent>
            <xs:extension base="samlp:RequestAbstractType">
                <xs:sequence>
                    <xs:element ref="xacml-context:Request"/>
                </xs:sequence>
                <xs:attribute name="InputContextOnly"
                    type="boolean"
                    use="optional"
                    default="false"/>
                <xs:attribute name="ReturnContext"
                    type="boolean"

```



```

                use="optional"
                default="false"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="XACMLPolicyQuery"
            type="XACMLPolicyQueryType"/>
<xs:complexType name="XACMLPolicyQueryType">
    <xs:complexContent>
        <xs:extension base="sampl:RequestAbstractType">
            <xs:choice minOccurs="0" maxOccurs="unbounded">>
                <xs:element ref="xacml-context:Request"/>
                <xs:element ref="xacml:Target"/>
                <xs:element ref="xacml:PolicySetIdReference"/>
                <xs:element ref="xacml:PolicyIdReference"/>
            </xs:choice>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
</schema>

```

D.4 XACML SAML 声明模式定义

本部分提供 XACML SAML 声明模式定义。

```

<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="urn:oasis:xacml:2.0:saml:assertion:schema:os"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:sampl="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
    xmlns:xacml="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified"
    blockDefault="substitution"
    version="2.0">
    <xs:import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
        schemaLocation="http://www.oasis-
open.org/committees/tc_home.php?wg_abbrev=security"/>
    <xs:import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
        schemaLocation="http://www.oasis-
open.org/committees/tc_home.php?wg_abbrev=security"/>
    <xs:import namespace="urn:oasis:names:tc:xacml:2.0:context:schema:os"
        schemaLocation="http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-
context-schema-os.xsd"/>
    <xs:import namespace="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
        schemaLocation="http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-
policy-schema-os.xsd"/>
    <xs:annotation>
        <xs:documentation>
            Document identifier: access_control-xacml-2.0-saml-assertion-schema-cd-02.xsd
            Location: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-
assertion-schema-cd-os.xsd
        </xs:documentation>
    </xs:annotation>
    <!-- -->
    <xs:element name="XACMLAuthzDecisionStatement"
        type="XACMLAuthzDecisionStatementType"/>
    <xs:complexType name="XACMLAuthzDecisionStatementType">
        <xs:complexContent>
            <xs:extension base="sampl:StatementAbstractType">
                <xs:sequence>
                    <xs:element ref="xacml-context:Response"/>
                    <xs:element ref="xacml-context:Request" MinOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

```

```

</xs:complexType>
  <!-- -->
  <xs:element name="XACMLPolicyStatement"
    type="XACMLPolicyStatementType"/>
  <xs:complexType name="XACMLPolicyStatementType">
    <xs:complexContent>
      <xs:extension base="sampl:StatementAbstractType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">>
          <xs:element ref="xacml:Policy"/>
          <xs:element ref="xacml:PolicySet"/>
        </xs:choice>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</schema>

```

附 录 一

安全考虑

本附录标识在实现基于 XACML 的系统时应当考虑的可能的安全和秘密折中场景。决定这些折中场景在他们的环境中是否是实际的,以及选择合适的安全措施留待执行者来确定。

I.1 威胁模型

我们此处假定对手已经接入到 XACML 动作者之间的通信信道并能够解释、插入、删除和修改消息或部分消息。

另外,一个动作者可以在后续的事物处理中使用来自前者恶意的消息。进一步假定规则和策略仅与生成和使用它们的动作者一样可靠。那么,每一个动作者有责任建立它依赖的其他动作者的合适的信任。信任建立机制不在本建议书的范围内。

在 XACML 模型的动作者之间传输的消息容易受到有恶意的第三方的攻击。攻击的其他点包括 PEP、PDP 和 PAP。严格来讲,这些实体中的一些不在本建议书的范围内,它们的安全将导致 PEP 所实施的接入控制的安全。

应当注意分布式系统中的其他构件,如在威胁模型的不在该讨论范围内的操作系统和域名系统(DNS)可能受到安全威胁。这些构件中的安全也可以导致策略侵害。

下列各节详细描述可能与 XACML 系统相关的特定安全场景。

I.1.1 未授权的暴露

XACML 不规定保护动作者之间互换的消息的机密性的任何固有的机制。因而,敌手可能观察在传输中的消息。在特定的安全策略下,该信息的暴露是有侵害的。主体提交的决定请求的属性和类型可能是对私有策略的破坏。在商务方面,对于个人数据的未授权的泄露的后果可以是管理者的受窘到监禁,而对与医学或金融数据而言,其后果则是大量的罚金。

采用机密性的安全措施来解决未授权的泄露。

I.1.2 消息重放

消息重放攻击是敌手在 XACML 动作者之间录制和重放合法信息的一种情况。该冲击可以导致服务的否认、过时信息或扮演信息的使用。

防止重放攻击需要使用消息全新的安全措施。

注意由于消息仅简单地重放并不需要由敌方来明白,消息加密没有减轻重放攻击。

I.1.3 消息插入

消息插入攻击是一种敌方在 XACML 动作者之间的消息序列中插入消息的一种情况。

防止消息插入攻击的一种解决方案是在动作者之间使用相互认证和消息序列完整性安全保护。应当注意仅使用 SSL 相互认证是不够的。相互认证仅证明另一方是由 X.509 证书的主体所标识的一方。为了更加有效，需要证实证书主体被授权发送消息。

I.1.4 消息删除

消息删除攻击是指对方在 XACML 动作者之间删除消息序列中的消息。消息删除可能导致服务的否认。然而，适当地设计的 XACML 系统不将不正确的认证决定作为消息删除冲击的结果。

防止消息删除攻击的解决方案是在动作者之间使用消息序列完整性保护。

I.1.5 消息修改

若对方能截取消息并改变其内容，那么，他们能够改变授权的决定。消息完整性保护可以阻止成功的消息修改冲击。

I.1.6 NotApplicable 的结果

“NotApplicable”的结果意味着 PDP 不能定位其目的匹配决定请求的策略位置。一般来说，强烈建议使用“Deny”结果策略，这样，在 PDP 要返回“NotApplicable”的时候，返回“Deny”作为替代结果。

然而，在某些安全模型中，象在许多 Web 服务器中发现的那些，“NotApplicable”的授权决定被作为等同于“Permit”处理。为安全起见，有一些必须加以考虑的特定安全事项。在下面一些段落中对这些进行解释。

若不将“NotApplicable”作为“Permit”来处理，必须将在决定请求中匹配单元由该策略使用的匹配算法紧紧地与将提交到决定请求的应用使用的数据句法相对准。匹配的失败将导致“NotApplicable”并作为“Permit”来对待。因而，未期望的匹配的失败可能允许未期望的接入。

商用的 http 响应允许同等处理各种句法。“%”可以被用做表示十六进制值字符。URL 路径“/./”提供规定相同值的多种方法。可以允许多种字符集，在某些情况下，相同的打印字符可能采用不同的二进制值表示。除非策略使用的匹配算法能够满足这些变化，不希望的接入可能会得到允许。

只有在保证明确地表示决定请求的所有应用使用策略所希望的精确句法的闭合环境中，将“NotApplicable”作为“Permit”来对待才是安全的。在更加开放的环境中，可以从使用任何合法的句法的应用处接收到决定请求，除非不管句法或类型变化，匹配规则已经被精心设计成匹配所有可能的应用输入，强烈建议不将“NotApplicable”作为“Permit”处理。除非 PEP 接收到一个显式的“Permit”授权决定，PEP 必须拒绝接入。

I.1.7 否定规则

否定规则是基于不是“True”的判断。若不精心使用，否定规则能导致策略侵害，因而一些权威建议不使用否定规则。然而，否定规则在特定场合相当有效，这样 XACML 选择包括否定规则。不过，建议小心使用否定规则，若可能尽可能避免使用否定规则。

否定规则的通常用法是在较大组中的成员允许他们接入的情况下，拒绝接入到某个个体或子组。例如，我们可能想要写一个规则，除了 Joe（他是唯一的一个礼仪副总并可能在其交谈过程时不慎重），允许所有的副总看未发布的商务数据。若我们完全控制了主体属性的管理，较好的方法将是将“副总”和“礼仪副总”规定为截然不同的组，然后再规定规则。然而，在某些环境下，这种方法可能不灵活。（一般而言，值得注意在这一点上在规则中涉及到个体不能很好地界定。一般而言，首选共享属性。）

如果使用不当，否定规则可能在两种共同的情况下导致违反策略。它们是：在属性被抑制时以及基本组改变时。属性被抑制的一个例子是若我们有一个除非主体是信用风险，允许接入的策略。若由于某种原因，PDP 可能不知道信用风险的属性，那么可能导致未授权的接入。在某些环境下，主体可能通过私有控制的应用，或包含有信息的服务器或知识库由于意外或主观原因无效，来抑制属性的公布。

改变基本组的例子将是:若有一个策略,该策略是工程部除了秘书外的每一个人可以改变软件源代码。假定目前该部将与另外一个工程部合并,同时想要保留该策略。然而,新部也包括标识为管理助理的个体,这些助理应当以同样的方式被作为秘书来看待。除非改变了策略,他们将无意地被允许改变软件源代码。在一个个体管理所有的策略时,该类问题很容易避免,但是当管理被分散时,按 XACML 允许,这类情况必须明确地预防。

I.2 安全措施

I.2.1 认证

认证提供事物处理中的一方确定事物处理中的另一方身份的方法。认证可以是一个方向或是双边的。

给定接入控制系统的敏感特性,对于 PEP 而言,对它向其发送决定请求的 PDP 身份进行认证是重要的。否则,将有对手可能提供假的或无效的授权决定从而导致策略被侵害的危险。

对 PDP 而言,认证 PEP 的身份并评估信任等级来确定什么样的敏感数据(若有)能够通过同等重要。应当牢记若对手被允许向 PDP 进行无限制的请求,可以使用简单的“Permit”或“Deny”。

许多不同的技术可以用做提供认证,像协同定位码、专用网络、VPN 或数字签名。认证也可以作为用于互换上下文的通信协议的一部分。在这种情况下,认证可以在消息级或在会话级上执行。

I.2.2 策略管理

若策略的内容暴露在接入控制系统外面,潜在的主体可以使用该信息确定如何获得未授权的接入。

为避免该威胁,用做存储策略的知识库可能本身需要接入控制,另外,<Status>单元应用做返回仅在那些属性的身份暴露将不对安全造成威胁时返回丢失的属性值。

I.2.3 机密性

机密性机制确保消息的内容仅能由希望得到的接收者阅读而不被在消息传输过程中遇见到该消息的任何其他人阅读。在机密性中有两个领域应加以考虑:一个是在传输过程中的机密性;另一个是在<Policy>单元中的机密性。

I.2.3.1 通信机密性

在某些环境中,将接入控制系统中的所有数据均看作是秘密的确是好习惯。在其他环境中,策略将对分发、检查和审计随时有效。将保持策略信息保密的想法是使得策略信息对对方而言更难以知道通过什么步骤才能足够获得未授权的接入。不管选择什么样的方法,接入控制系统的安全不应依赖于策略的安全。

任何与传输或互换 XACML<Policy>单元相关的安全考虑不在 XACML 标准的范围内。当在两方之间进行互换时保留<Policy>单元的完整性和机密性通常很重要的情况下,由实施者来确定对于他们的环境的合适的机制。

通信机密性可以通过机密性机制,如 SSL 来提供。使用像 SSL 的点到点方案在一个端点有安全危险时,可能导致其他的攻击。

I.2.3.2 陈述级机密性

在某些场合,实施可能想加密 XACML <Policy>单元的部分内容。

W3C 加密法:2002 可能被用做加密所有或部分 XML 文档。建议将 W3C 加密法:2002 用于 XACML。

不言而喻，若知识库被用做方便在 PAP 和 PDP 之间明文（例如，未加密的）策略的通信，那么安全知识库将用于存储该敏感数据。

I.2.4 策略完整性

PDP 使用 XACML 策略来赋值请求上下文的 XACML 策略是该系统的核心。因而，保持其完整性是必要的。从两个方面来保持该策略的完整性。一个是确保<Policy>单元自最初由 PAP 创建开始不被替换。另一个是确保<Policy>单元没有从策略集中被插入或删除。

在许多情况下，两个方面均能够通过确保动作者的完整性以及为保护动作者之间的通信执行会话级机制来实现。然而，当在后来的时刻动作的组织之间进行分发时或在该策略携带保护的资源进行传递时，签署该策略是有用的。在这些情况下，建议将 W3C 的 XML 签名句法和处理标准用于 XACML。

数字签名应仅用于确保该声明的完整性。数字签名不应用做选择或赋值策略的一种方法。也就是说，PDP 不应根据谁签署了它或者它是否已经被签署或没有被签署来请求策略（选择的确基础本身是策略的问题）。然而，PDP 必须验证签署使用的密钥是策略授权的签署者控制的密钥。进行签署的方法取决于选择的特定签名技术同时不在本建议书的范围内。

I.2.5 策略标识符

由于策略可能通过它们的标识符来引用，确保这些标识符的唯一性是 PAP 的责任。标识符之间的混淆可能导致该可应用策略的错误标识。本建议书对在一个策略被修改或在被修改的策略中可以使用相同的标识符 PAP 是否必须生成一个新标识符不进行规定。这是管理者在实施时的问题。然而，对任一种情况均必须注意。若标识符被重复使用，将有引用它的其他策略或策略集可能会受到影响的危险。相反地，若使用一个新的标识符，除非先前的策略被删除，这些引用其他策略可能继续使用先前的策略。在任一种情况下，结果均不是策略管理者希望的。

I.2.6 信任模型

认证、完整性和机密性安全措施的讨论有必要假定底层信任模型：如何使一个动作者能够相信给定的密钥唯一地与一个特定的、标识的动作者相关联，以致于该密钥可以使那个动作者用作加密数据或验证来自那个动作者的签名（或其他完整性结构）？存在许多不同类型的信任模型，包括严格的体系结构、分布的授权机构、Web、桥等等。

按照存在和不存在的相互依赖关系，值得考虑接入控制系统的各种动作者之间的关系。

- 没有任何授权系统的实体依赖于 PEP。它们可以从它收集数据，例如，认证数据，并且它们自身需要负责验证它（PEP）。
- 为实际执行策略决定，该系统的正确操作与 PEP 的能力无关。
- PEP 依赖于 PDP 进行正确地赋值策略。为实际上执行策略决定，系统的正确操作依赖于 PEP 的能力。这依次暗示为 PDP 提供了正确输入。除此之外，PDP 不依赖于 PEP。
- PAP 依赖 PDP 提供适当的策略。PAP 不依赖其他构件。

I.2.7 秘密

意识到与相关的接入控制相关的任何事物处理可能暴露有关动作者的私有信息是重要的。例如，若 XACML 策略说明特定数据仅可以由具有“金卡会员”状态的主体阅读，那么允许主体接入那类数据的任何事物处理向对手泄露有关主体的状态。这样秘密考虑可以导致围绕 XACML 策略场景执行本身加密和/或接入控制需求：为请求/响应协议消息受保护的信任信道，存储和传递中主体属性的保护等等。

适合于给定环境秘密机制的选择和使用不在 XACML 的范围内。有关是否、如何以及何时开发这类机制的相关决定留给与该环境相关的实施者。

附录二

XACML示例

本附录包含两个用于说明性目的的 XACML 使用的示例。第一个示例是列举目标、上下文、匹配功能和主体属性的使用的简单示例。第二个示例附加示出组合规则算法、条件和职责规则的使用。

II.1 示例一

本部分包含第一个示例。

II.1.1 示例策略

假定一个名称为 Medi Corp（用域名标识：med.example.com）具有用英文表示的一个接入控制策略：

任何具有“med.example.com”命名空间中的一个 e-mail 名称的用户允许针对任何资源执行任何动作。

XACML 策略由头信息、该策略的选用的文本描述、目标、一个或多个规则以及一个职责的选用集组成。

```
[a02] <?xml version="1.0" encoding="UTF-8"?>
[a03] <Policy
[a04]   xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
[a05]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[a06]   xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-
os.xsd"
[a07]   PolicyId="urn:oasis:names:tc:example:SimplePolicy1"
[a08]   RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-
overrides">
[a09]   <Description>
[a10]     Medi Corp access control policy
[a11]   </Description>
[a12]   <Target/>
[a13]   <Rule
[a14]     RuleId="urn:oasis:names:tc:xacml:2.0:example:SimpleRule1"
[a15]     Effect="Permit">
[a16]     <Description>
[a17]       Any subject with an e-mail name in the med.example.com domain
[a18]       can perform any action on any resource.
[a19]     </Description>
[a20]     <Target>
[a21]       <Subjects>
[a22]         <Subject>
[a23]           <SubjectMatch
[a24]             MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-
match">
[a25]             <AttributeValue
[a26]               DataType="http://www.w3.org/2001/XMLSchema#string">
[a27]               med.example.com
[a28]             </AttributeValue>
[a29]             <SubjectAttributeDesignator
[a30]               AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[a31]               DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
[a32]             </SubjectMatch>
[a33]           </Subject>
[a34]         </Subjects>
[a35]       </Target>
[a36]     </Rule>
[a37]   </Policy>
```

[a02]是指示使用 XML 的哪个版本以及字符编码是什么的标准 XML 文本标签。

[a03]介绍 XACML 策略本身。

[a04] – [a05]是 XML 命名空间声明。

[a04]给出 XACML 策略方案的一个 URN。

[a07]为策略场合分配一个名字。对于一个给定的 PDP，策略名字必须是唯一的，这样在另一个策略引用一个策略时没有歧义。省略版本属性，取其缺省值“1.0”。

[a08]规定将被用做解析可能在该策略的各种规则结果的算法。此处规定的拒绝控制组合规则算法表示，若任何一个规则赋值为“Deny”，那么该策略必须返回“Deny”。若所有规则赋值为“Permit”，那么该策略必须返回“Permit”。在附录 C 详细描述的组合规则算法也表示在赋值任何规则时若出现差错该做什么以及对不应用于特定决定请求的规则应做什么。

[a09] – [a11]提供该策略的文本描述。该描述是可选的。

[a12] 描述该策略应用的决定请求。若一个决定中的主体、资源、动作以及环境与该策略目标中规定的值不匹配，那么不需要赋值该策略的剩余部分。该目标部分对创建策略集的一个指示是有用的。在该简单示例中，目标部分表示该策略可应用于任何决定请求。

[a13]介绍该简单策略中的一个且唯一规则。

[a14] 规定该规则的标识符。正如用于一个策略，每一个规则必须具有唯一标识符（至少对将使用该策略的任何 PDP 是唯一的。）

[a15] 表示若规则赋值为“True”，该规则具有什么效果。规则可以有“Permit”或“Deny”的效果。在该种情况下，若该规则是满足的，将赋值为“Permit”，含义是，只要涉及到该规则，请求的接入应被允许。若规则赋值是“False”，那么返回“NotApplicable”结果。若在赋值该规则时出现差错，那么该规则返回“Indeterminate”。如上所述，用于该策略的组合规则算法规定如何将各种规则值组合成一个单一的策略值。

[a16] – [a19]提供了该规则的文字描述。该描述是选用的。

[a20]介绍规则的目标。如上面对策略的目标的描述，规则的目标描述该规则应用的决定请求。若决定请求中的主体、资源、动作和环境与规则目标中规定的值不匹配，那么不需要赋值规则的剩余部分，对该规则赋值返回“NotApplicable”的值。

规则目标类似于策略目标本身，但有一个重要的不同。

[a23] – [a32]拼写出决定请求中的主体必须匹配的特定值。<SubjectMatch>单元规定 MatchId 属性、“med.example.com”的文字值和通过<SubjectAttributeDesignator>单元在请求上下文中的特定主体属性的一个指针的匹配功能。该匹配功能将被用做比较文字值与具有主体属性的值。当且仅当匹配返回“True”时，该规则应用于特定的决定请求。若该匹配返回“False”，那么该规则将返回值“NotApplicable”。

[a36]关闭规则。在该规则中，所有的工作在<Target>单元中完成。在更复杂的规则中，<Target>可能后随<Condition>单元（也可以是 ANDed 或 Ored 条件集）。

[a37]关闭该策略。如上所述，该策略仅有一个规则，但更复杂的策略可以有一些规则。

II.1.2 示例请求上下文

检查可能提交到执行上面策略的 PDP 的假定决定请求。以英文表示的生成决定请求的接入请求可以陈述如下：

具有 E-mail 名称为 bs@simpsons.com 的 Bart Simpsom，可在 Medi Corp 上读到他的医学病历。

以 XACML 表示的决定请求中的信息格式化为请求上下文声明如下：

```
[a38] <?xml version="1.0" encoding="UTF-8"?>
[a39] <Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[a40]   xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
http://docs.oasis-
open.org/xacml/access_control-xacml-2.0-context-schema-os.xsd">
[a41]   <Subject>
[a42]     <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
[a43]       <AttributeValue>
[a44]         bs@simpsons.com
[a45]       </AttributeValue>
[a46]     </Attribute>
[a47]   </Subject>
[a48]   <Resource>
[a49]     <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#anyURI">
[a50]       <AttributeValue>
[a51]         file://example/med/record/patient/BartSimpson
[a52]       </AttributeValue>
[a53]     </Attribute>
[a54]   </Resource>
[a55]   <Action>
[a56]     <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" DataType="http://www.w3.org/2001/XMLSchema#string">
[a57]       <AttributeValue>
[a58]         read
[a59]       </AttributeValue>
[a60]     </Attribute>
[a61]   </Action>
[a62] </Environment/>
[a63] </Request>
```

[a38] – [a40]包含请求上下文头信息，并与上面解释的策略的头相同方式使用。

<Subject>单元包含进行接入请求的实体的一个或多个属性。可能有多个主体，每一个主体可能有多个属性。在该种情况下，在[a41] – [a47]中，仅有一个主体，该主体仅有一个属性：采用 e-mail 名字表示的主体的身份是 bs@simpsons.com。在本示例中，subject-category 属性可以省略。因而，它采用“接入主体”的缺省值。

<Resource>单元包含主体已经请求接入的资源的一个或多个属性。每一个决定请求可能仅有一个 <Resource>。行[a48] – [a54]包含 Bart Simpson 已经请求接入的资源的一个属性：由文件 URI 标识的资源，即：“file://medico/record/patient/BartSimpson”。

<Action>单元包含主体希望使用资源的动作一个或多个属性。每一个决定请求可能仅有一个动作。[a55] – [a61]描述了 Bart Simpson 希望进行的动作（读动作）的身份。

<Environment>单元[a62]是空。

[a63]关闭请求上下文。更为复杂的请求上下文可能已经包含一些与主体、资源或动作无关联的某些属性。这些已经放置在<Action>单元后面的可选<Environment>单元中。

处理该请求上下文的 PDP 将该策略定位于其策略知识库中。它将请求上下文中所包含的主体、资源、动作和环境与策略目标中的主体、资源、动作和环境进行比较。由于策略目标是空，策略匹配该上下文。

PDP 目前将请求上下文中的主体、资源、动作和环境与该策略中一个规则的目标相比较。请求的资源匹配 <Target>单元，同时请求的动作匹配 <Target>单元，但请求“subject-id 属性不匹配“med.example.com”。

II.1.3 示例响应上下文

作为赋值策略的结果，为该请求返回“Permit”结果的该策略中没有规则。在此种情况下，用于策略的组合规则算法规定应返回“NotApplicable”结果。响应上下文如下：

```
[a64] <?xml version="1.0" encoding="UTF-8"?>
[a65] <Response xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
http://docs.oasis-open.org/xacml/xacml-core-2.0-context-schema-os.xsd">
[a66] <Result>
[a67] <Decision>NotApplicable</Decision>
[a68] </Result>
[a69] </Response>
```

[a64] – [a65]对响应包含上面描述的对策略相同种类头信息。

在行[a66] – [a68]中的<Result>单元包含针对该策略赋值决定请求的结果。在这种情况下，结果是“NotApplicable”。一个策略可能返回“Permit”、“Deny”、“NotApplicable”或“Indeterminate”。因而，需要 PEP 拒绝接入。

[a69]关闭了响应上下文。

II.2 示例二

本部分包含一个典型的 XML 文档，一个典型的请求上下文和典型的 XACML 规则。XML 文档是病历档案。规定了四个单独的规则。这些示出组合规则算法、条件和职责。

II.2.1 示例的病历档案情况

下面是示例 XACML 规则可应用的病历档案的情况。在由 Medi Corp 管理的注册命名空间中规定了<record>方案。

```
[a70] <?xml version="1.0" encoding="UTF-8"?>
[a71] <record xmlns="urn:example:med:schemas:record"
[a72] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
[a73] <patient>
[a74] <patientName>
[a75] <first>Bartholomew</first>
[a76] <last>Simpson</last>
[a77] </patientName>
[a78] <patientContact>
[a79] <street>27 Shelbyville Road</street>
[a80] <city>Springfield</city>
[a81] <state>MA</state>
[a82] <zip>12345</zip>
[a83] <phone>555.123.4567</phone>
[a84] <fax/>
[a85] <email/>
[a86] </patientContact>
[a87] <patientDoB>1992-03-21</patientDoB>
[a88] <patientGender>male</patientGender>
[a89] <patient-number>555555</patient-number>
[a90] </patient>
[a91] <parentGuardian>
[a92] <parentGuardianId>HS001</parentGuardianId>
[a93] <parentGuardianName>
[a94] <first>Homer</first>
[a95] <last>Simpson</last>
[a96] </parentGuardianName>
[a97] <parentGuardianContact>
[a98] <street>27 Shelbyville Road</street>
[a99] <city>Springfield</city>
[a100] <state>MA</state>
[a101] <zip>12345</zip>
[a102] <phone>555.123.4567</phone>
[a103] <fax/>
[a104] <email>homers@aol.com</email>
```

```

[a105] </parentGuardianContact>
[a106] </parentGuardian>
[a107] <primaryCarePhysician>
[a108] <physicianName>
[a109] <first>Julius</first>
[a110] <last>Hibbert</last>
[a111] </physicianName>
[a112] <physicianContact>
[a113] <street>1 First St</street>
[a114] <city>Springfield</city>
[a115] <state>MA</state>
[a116] <zip>12345</zip>
[a117] <phone>555.123.9012</phone>
[a118] <fax>555.123.9013</fax>
[a119] <email/>
[a120] </physicianContact>
[a121] <registrationID>ABC123</registrationID>
[a122] </primaryCarePhysician>
[a123] <insurer>
[a124] <name>Blue Cross</name>
[a125] <street>1234 Main St</street>
[a126] <city>Springfield</city>
[a127] <state>MA</state>
[a128] <zip>12345</zip>
[a129] <phone>555.123.5678</phone>
[a130] <fax>555.123.5679</fax>
[a131] <email/>
[a132] </insurer>
[a133] <medical>
[a134] <treatment>
[a135] <drug>
[a136] <name>methylphenidate hydrochloride</name>
[a137] <dailyDosage>30mgs</dailyDosage>
[a138] <startDate>1999-01-12</startDate>
[a139] </drug>
[a140] <comment>
[a141] patient exhibits side-effects of skin coloration and carpal
degeneration
[a142] </comment>
[a143] </treatment>
[a144] <result>
[a145] <test>blood pressure</test>
[a146] <value>120/80</value>
[a147] <date>2001-06-09</date>
[a148] <performedBy>Nurse Betty</performedBy>
[a149] </result>
[a150] </medical>
[a151] </record>

```

II.2.2 典型的请求上下文

下面的示例给出了示例型规则可应用的请求上下文。它表示医师 Julius Hibbert 要在 Bartholomew Simpson 的档案中看患者出生日期的请求。

```

[a152] <?xml version="1.0" encoding="UTF-8"?>
[a153] <Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-
os.xsd">
[a154] <Subject>
[a155] <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject-
category"
DataType="http://www.w3.org/2001/XMLSchema#anyURI">
[a156] <AttributeValue>urn:oasis:names:tc:xacml:1.0:subject-
category:access-subject</AttributeValue>
[a157] </Attribute>
[a158] <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="med.example.com">

```

```

[a159]   <AttributeValue>CN=Julius Hibbert</AttributeValue>
[a160]   </Attribute>
[a161]   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:name-
format"
DataType="http://www.w3.org/2001/XMLSchema#anyURI"
Issuer="med.example.com">
[a162]   <AttributeValue>
[a163]     urn:oasis:names:tc:xacml:1.0:datatype:x500name
[a164]   </AttributeValue>
[a165] </Attribute>
[a166] <Attribute
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="med.example.com">
[a167]   <AttributeValue>physician</AttributeValue>
[a168] </Attribute>
[a169] <Attribute
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:physician-id"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="med.example.com">
[a170]   <AttributeValue>jh1234</AttributeValue>
[a171] </Attribute>
[a172] </Subject>
[a173] <Resource>
[a174]   <ResourceContent>
[a175]     <md:record xmlns:md="urn:example:med:schemas:record"
xsi:schemaLocation="urn:example:med:schemas:record
http:www.med.example.com/schemas/record.xsd">
[a176]       <md:patient>
[a177]         <md:patientDoB>1992-03-21</md:patientDoB>
[a178]         <md:patient-number>555555</md:patient-number>
[a179]       </md:patient>
[a180]     </md:record>
[a181]   </ResourceContent>
[a182] </Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string">
[a183]   <AttributeValue>
[a184]     //med.example.com/records/bart-simpson.xml#
[a185] xmlns(md=:Resource/ResourceContent/xpointer
[a186] (/md:record/md:patient/md:patientDoB)
[a187]   </AttributeValue>
[a188] </Attribute>
[a189] </Resource>
[a190] <Action>
[a191] <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id"
DataType="http://www.w3.org/2001/XMLSchema#string">
[a192]   <AttributeValue>read</AttributeValue>
[a193] </Attribute>
[a194] </Action>
[a195] <Environment/>
[a196] </Request>

```

[a152] – [a153]标准命名空间声明。

[a154] – [a172]将主体属性放在<Request>单元的<Subject>单元中。每一个属性由属性元数据和属性值组成。在该请求中仅包含一个主体。

[a155] – [a157]每一个<Subject>单元有一个 SubjectCategory 属性。该属性值描述相关主体在进行决定请求中所起的作用。"access-subject"的值表示为其签署请求的身份。

[a158] – [a160]主体 subject-id 属性。

[a161] – [a165]subject-id 的格式。

[a166] – [a168]主体 role 属性。

[a169] – [a171]主体 physician-id 属性。

[a173]–[a189]将资源属性放置在<Request>单元的<Resource>单元中。每一个属性由属性元数据和属性值组成。

[a174]–[a181]资源内容。此处存放可能请求接入的全部或部分 XML 资源情况。

[a182] – [a188] 请求接入的资源情况标识符，该标识符是 Xpath 表达成选择要接入的数据的 <ResourceContent>单元。

[a190]–[a194]在<Request>单元的<Action>单元中存放的动作属性。

[a192]动作标识符。

[a195]空<Environment>单元。

II.2.3 示例普通语言规则

要执行下面的普通语言规则：

- 1) 规则1：采用其患者号标识的一个人可以看任何他是被指派的患者的档案。
- 2) 规则2：若某人是患者的父母或监护人同时患者的年龄不足16岁，那么此人可以看该患者的所有病历。
- 3) 规则3：若邮件发送到一个患者，那么该患者的主治医师可以为其写任何病历内容。
- 4) 规则4：管理者不允许看或写任何病历档案中任何病历内容。

这些规则可以由独立运行的不同 PAP 或由单一的 PAP 来写。

II.2.4 示例XACML规则情况

II.2.4.1 规则1

规则 1 示出了具有单一<Condition>单元的简单规则。它也给出了规定在策略中可以使用的功能的 <VariableDefinition>单元的使用。

下列 XACML <Rule>情况表示规则 1：

```
[a197] <?xml version="1.0" encoding="UTF-8"?>
[a198] <Policy
[a199] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
[a200] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=" urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-
os.xsd"
[a201] xmlns:md="http://www.med.example.com/schemas/record.xsd"
[a202] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:1"
[a203] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:deny-overrides">
[a204] <PolicyDefaults>
[a205] <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
19991116</XPathVersion>
[a206] </PolicyDefaults>
[a207] <Target/>
[a208] <VariableDefinition VariableId="17590034">
[a209] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
[a210] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
one-and-only">
[a211] <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:patient-number"
[a212] DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a213] </Apply>
[a214] <Apply
[a215] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-
only">
[a216] <AttributeSelector
[a217] RequestContextPath="//xacml-context:Resource/xacml-
context:ResourceContent/md:record/md:patient/md:patient-number/text () "
```

```

[a218]     DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a219]     </Apply>
[a220]   </Apply>
[a221] </VariableDefinition>
[a222] <Rule
[a223] RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1"
[a224] Effect="Permit">
[a225]   <Description>
[a226]     A person may read any medical record in the
[a227]     http://www.med.example.com/schemas/record.xsd namespace
[a228]     for which he or she is the designated patient
[a229]   </Description>
[a230]   <Target>
[a231]     <Resources>
[a232]       <Resource>
[a233]         <ResourceMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a234]           <AttributeValue
[a235]             DataType="http://www.w3.org/2001/XMLSchema#string">
[a236]               urn:example:med:schemas:record
[a237]             </AttributeValue>
[a238]           <ResourceAttributeDesignator AttributeId=
[a239]             "urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
[a240]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a241]           </ResourceMatch>
[a242]         <ResourceMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
[a243]           <AttributeValue
[a244]             DataType="http://www.w3.org/2001/XMLSchema#string">
[a245]               /md:record
[a246]             </AttributeValue>
[a247]           <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
[a248]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a249]           </ResourceMatch>
[a250]         </Resource>
[a251]       </Resources>
[a252]     </Target>
[a253]     <Actions>
[a254]       <Action>
[a255]         <ActionMatch
[a256]           MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a257]             <AttributeValue
[a258]               DataType="http://www.w3.org/2001/XMLSchema#string">
[a259]                 read
[a260]               </AttributeValue>
[a261]             <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[a262]               DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a263]             </ActionMatch>
[a264]           </Action>
[a265]         </Actions>
[a266]       </Target>
[a267]       <Condition>
[a268]         <VariableReference VariableId="17590034"/>
[a269]       </Condition>
[a270]     </Rule>
[a271] </Policy>

```

[a199] – [a201]XML 命名空间声明。

[a205]策略中 Xpath 表达式将按照 W3C Xpath: 1999 来解释。

[a208] – [a221]一个<VariableDefinition>单元。它规定赋值陈述的正确性功能：patient-number 主体属性等同于资源中的 patient-number。

[a209]FunctionId 属性为用于比较的功能命名。在这种情况下，采用"urn:oasis:names:tc:xacml:1.0:function:string-equal"功能来完成比较；该功能使用类型"http://www.w3.org/2001/XMLSchema#string"的两个变量。

[a210]可变定义的第一个变量是由 FunctionId 属性规定的一个功能。由于 urn:oasis:names:tc:xacml:1.0:function:string-equal 采用了类型"http://www.w3.org/2001/XMLSchema#string"的变量同时 SubjectAttributeDesignator 选择了类型"http://www.w3.org/2001/XMLSchema#string"的一个包，因而使用了"urn:oasis:names:tc:xacml:1.0:function:string-one-and-only"。

该功能确保其变量赋值仅包含一个值的一个包。

[a211]SubjectAttributeDesignator 为请求上下文中的 patient-number 主体属性选择了一个包。

[a215]可定义的第二个变量是由 FunctionId 属性规定的一个功能。由于 urn:oasis:names:tc:xacml:1.0:function:string-equal 采用了类型 http://www.w3.org/2001/XMLSchema#string 的变量同时 AttributeSelector 选择了类型 "http://www.w3.org/2001/XMLSchema#string" 的一个包，因而使用了 "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only"。该功能确保其变量赋值仅包含一个值的一个包。

[a216]<AttributeSelector>单元从使用选择自由形态的 Xpath 表达式的请求上下文选择一包值。在该情况下，它选择了资源中的 patient-number 值。注意在 Xpath 表达式中的命名空间前缀采用标准的 XML 命名空间声明来解析。

[a223]规则标识符。

[a224]规则影响声明。当一个规则赋值为“True”时，它发出 Effect 属性值。该值然后按照组合规则算法与其他规则的 Effect 值组合。

[a225] – [a229]规则的自由形态描述。

[a230] – [a263]规则目标规定规则想要赋值的决定请求集。在该示例中，省略<Subjects>和<Environments>单元。

[a231] – [a249]<Resources>单元包含离散的<Resource>单元序列。在该示例中，只有一个单元。

[a232] – [a248]<Resources>单元包含 ResourceMatch 相连的单元序列。在该示例中，有两个单元。

[a233] – [a240]按照匹配功能，第一个<ResourceMatch>将其第一个和第二个子单元进行比较。若第一个变量值与第二个变量所选择的任一个值相匹配，该匹配是正。该匹配将请求的文档的目标命名空间与 "urn:example:med:schemas:record" 的值相比较。

[a233]MatchId 属性为匹配功能命名。

[a235]匹配的文字属性值。

[a237] – [a239]<ResourceAttributeDesignator>单元从请求上下文中包含的资源选择目标命名空间。采用 AttributeId 来规定属性名。

[a241] – [a247]第二个<ResourceMatch>单元。该匹配比较两个 Xpath 表达式的结果。第二个 Xpath 表达式是到请求的 XML 单元的位置路径，而第一个 Xpath 表达式是"/md:record"的文字值。若请求的 XML 单元在 "/md:record"单元的下面，那么"xpath-node-match"功能赋值为“True”。

[a250] – [a262]<Actions>单元包含<Action>离散的单元序列。在这种情况下，只有一个<Action>单元。

[a251] – [a261]<Actions>单元包含<ActionMatch>相连的单元序列。在这种情况下，只有一个<ActionMatch>单元。

[a252] – [a260]<ActionMatch>按照匹配功能将其第一个和第二个子单元进行比较。若第一个变量值与第二个变量所选择的任一个值相匹配，该匹配是正。在该情况下，将请求上下文中的 action-id 动作属性值与文字值"read"相比较。

[a264] – [a266]<Condition>单元。对于可应用的规则，一个条件必须赋值为“True”。该条件包含在策略中其他位置上规定的可变定义的一个引用。

II.2.4.2 规则2

规则 2 给出了算术功能的使用，例如，用以计算患者的 16 岁生日的日期的具有 FunctionId "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration"的<Apply>单元。它也给出了具有 FunctionId "urn:oasis:names:tc:xacml:1.0:function:and"的判断表达式的使用。该示例有潜在在<Condition>单元中的一个功能，以及在<VariableDefinition>单元中引用的另一个单元。

```
[a269] <?xml version="1.0" encoding="UTF-8"?>
[a270] <Policy
[a271]   xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
[a272]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
[a273]   xmlns:xf="urn:oasis:names:tc:xacml:2.0:data-types"
[a274]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
[a275]   PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:2"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-
overrides">
[a276]   <PolicyDefaults>
[a277]     <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-19991116</XPathVersion>
[a278]   </PolicyDefaults>
[a279]   <Target/>
[a280]   <VariableDefinition VariableId="17590035">
[a281]     <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:date-less-or-
equal">
[a282]       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-
only">
[a283]         <EnvironmentAttributeDesignator
[a284]           AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"
[a285]           DataType="http://www.w3.org/2001/XMLSchema#date"/>
[a286]         </Apply>
[a287]       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-
yearMonthDuration">
[a288]         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-
only">
[a289]           <AttributeSelector RequestContextPath=
[a290]             "http://www.med.example.com/schemas/record.xsd/md:record/md:patient/md:patientDoB/text()"
[a291]           DataType="http://www.w3.org/2001/XMLSchema#date"/>
[a292]         </Apply>
[a293]         <AttributeValue
[a294]           DataType="urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration">
[a295]           <xf:dt-yearMonthDuration>
[a296]             P16Y
[a297]           </xf:dt-yearMonthDuration>
[a298]         </AttributeValue>
[a299]       </Apply>
[a300]     </Apply>
[a301]   </VariableDefinition>
[a302]   <Rule
[a303]     RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:2"
[a304]     Effect="Permit">
[a305]     <Description>
[a306]       A person may read any medical record in the
[a307]       http://www.med.example.com/records.xsd namespace
[a308]       for which he or she is the designated parent or guardian,
[a309]       and for which the patient is under 16 years of age
[a310]     </Description>
[a311]     <Target>
[a312]     <Resources>
[a313]     <Resource>
[a314]       <ResourceMatch
[a315]         MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a316]         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
[a317]           http://www.med.example.com/schemas/record.xsd
[a318]         </AttributeValue>
[a319]       <ResourceAttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
[a320]       DataType="http://www.w3.org/2001/XMLSchema#string"/>
```

```

[a321]     </ResourceMatch>
[a322]     <ResourceMatch
[a323]     MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
[a324]     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
[a325]         /md:record
[a326]     </AttributeValue>
[a327]     <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
[a328]     DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a329]     </ResourceMatch>
[a330] </Resource>
[a331] </Resources>
[a332] <Actions>
[a333] <Action>
[a334] <ActionMatch
[a335] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a336] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
[a337]     read
[a338] </AttributeValue>
[a339] <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0: action:action-id"
[a340]     DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a341] </ActionMatch>
[a342] </Action>
[a343] </Actions>
[a344] </Target>
[a345] <Condition>
[a346] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
[a347] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a348] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-
and-only">
[a349] <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:2.0: example:attribute:
[a350] parent-guardian-id"
[a351]     DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a352] </Apply>
[a353] <Apply
[a354]     FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
[a355] <AttributeSelector
[a356]     RequestContextPath="//xacml-context:Resource/xacml-
context:ResourceContent/md:record/md:parentGuardian/md:parentGuardianId/text()"
[a357]     DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a358] </Apply>
[a359] </Apply>
[a360] <VariableReference VariableId="17590035"/>
[a361] </Apply>
[a362] </Condition>
[a363] </Rule>
[a364] </Policy>

```

[a280] – [a301] <VariableDefinition>单元包含条件部分（例如，患者年龄是否低于 16 岁？）。若当前的日期小于患者的出生日期加上 16 计算出来的日期，该患者小于 16 岁。

[a281] – [a300]用"urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal"计算两个日期变量的差异。

[a282] – [a286]第一个日期变量使用"urn:oasis:names:tc:xacml:1.0:function:date-one-and-only"以确保其变量选择的一包值包含类型 <http://www.w3.org/2001/XMLSchema#date> 的一个值。

[a284]通过选择"urn:oasis:names:tc:xacml:1.0:environment:current-date"环境属性来赋值当前日期。

[a287]–[a299]通过在患者的出生日上增加 16 来，第二个日期变量使用 "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" 来计算患者 16 岁的生日。其变量的第一个是类型 "http://www.w3.org/2001/XMLSchema#date" 第二个是类型 "urn:oasis:names:tc:xacml:2.0:data-types:yearMonthDuration"。

[a289]<AttributeSelector>单元通过在资源内容上采用 Xpath 表达式选择患者的出生日。

[a293]–[a298]16 年的年月持续时间。

[a311]–[a344]规则声明和规则目标。见 II.4.2.4.1 中针对这些单元详细描述的规则 1。

[a345]–[a362]<Condition>单元。对于可应用的规则，条件必须赋值为“True”。该条件赋值陈述的正确性：请求者是指定的父母或监护人，同时患者不足 16 岁。它包含一个潜入的<Apply>单元以及一个引用的<VariableDefinition>单元。

[a346]条件使用 "urn:oasis:names:tc:xacml:1.0:function:and" 功能。这是使用一个或多个布尔变量（在该情况下是 2）并执行逻辑 "AND" 操作来计算表达式的真值的布尔函数。

[a347]–[a359]该条件的第一部分被赋值（例如，请求者是否是指定的父母或监护人？）。函数是 "urn:oasis:names:tc:xacml:1.0:function:string-equal" 同时该函数取类型 "http://www.w3.org/2001/XMLSchema#string" 的两个变量。

[a348]指定第一个变量。由于 "urn:oasis:names:tc:xacml:1.0:function:string-equal" 采用类型 "http://www.w3.org/2001/XMLSchema#string" 的变量，"urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" 被用于确保请求上下文中的 "urn:oasis:names:tc:xacml:2.0:example:attribute:parent-guardian-id" 只包含一个值。

[a353]指定第二个变量。使用<SubjectAttributeDesignator>单元从请求上下文选择主体属性 "urn:oasis:names:tc:xacml:2.0:example:attribute:parent-guardian-id"。

[a354]如上所述，使用 "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" 来确保其变量选择的值包只包含类型 "http://www.w3.org/2001/XMLSchema#string" 的一个值。

[a355]使用<AttributeSelector>单元，第二个变量从资源内容选择<md:parentGuardianId>值。该单元包含指向请求上下文的自由形式 Xpath 表达式。注意 Xpath 表达式中的所有命名空间前缀使用标准的命名空间声明来解析。AttributeSelector 赋值类型 "http://www.w3.org/2001/XMLSchema#string" 的值包。

[a360]引用<VariableDefinition>单元，此处，规定条件的第二部分。

II.2.4.3 规则3

规则 3 给出职责的使用。XACML <Rule>单元句法不包括适合于承载职责的单元，因而，规则 3 必须按<Policy>单元格式化。

```
[a365] <?xml version="1.0" encoding="UTF-8"?>
[a366] <Policy
[a367]   xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xacml-context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
[a368]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[a369]   xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-
os.xsd"
[a370]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
[a371]   PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:3"
[a372]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:deny-overrides">
[a373]   <Description>
[a374]     Policy for any medical record in the
[a375]     http://www.med.example.com/schemas/record.xsd namespace
[a376]   </Description>
[a377] </PolicyDefaults>
```

```

[a378] <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
19991116</XPathVersion>
[a379] </PolicyDefaults>
[a380] <Target>
[a381] <Resources>
[a382] <Resource>
[a383] <ResourceMatch
[a384]   MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a385]   <AttributeValue
DataTyPe="http://www.w3.org/2001/XMLSchema#string">
[a386]     urn:example:med:schemas:record
[a387]   </AttributeValue>
[a388]   <ResourceAttributeDesignator AttributeId=
[a389]     "urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
[a390]     DataTyPe="http://www.w3.org/2001/XMLSchema#string"/>
[a391]   </ResourceMatch>
[a392] </Resource>
[a393] </Resources>
[a394] </Target>
[a395] <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:3"
[a396] Effect="Permit">
[a397] <Description>
[a398]   A physician may write any medical element in a record
[a399]   for which he or she is the designated primary care
[a400]   physician, provided an email is sent to the patient
[a401] </Description>
[a402] <Target>
[a403] <Subjects>
[a404] <Subject>
[a405] <SubjectMatch
[a406]   MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a407]   <AttributeValue
DataTyPe="http://www.w3.org/2001/XMLSchema#string">
[a408]     physician
[a409]   </AttributeValue>
[a410]   <SubjectAttributeDesignator AttributeId=
"urn:oasis:names:tc:xacml: 2.0:example:attribute:role"
[a411]     DataTyPe="http://www.w3.org/2001/XMLSchema#string"/>
[a412]   </SubjectMatch>
[a413] </Subject>
[a414] </Subjects>
[a415] <Resources>
[a416] <Resource>
[a417] <ResourceMatch
[a418]   MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-
match">
[a419]   <AttributeValue
[a420]     DataTyPe="http://www.w3.org/2001/XMLSchema#string">
[a421]     /md:record/md:medical
[a422]   </AttributeValue>
[a423]   <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml: 1.0:resource:xpath"
[a424]     DataTyPe="http://www.w3.org/2001/XMLSchema#string"/>
[a425]   </ResourceMatch>
[a426] </Resource>
[a427] </Resources>
[a428] <Actions>
[a429] <Action>
[a430] <ActionMatch
[a431]   MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a432]   <AttributeValue
[a433]     DataTyPe="http://www.w3.org/2001/XMLSchema#string">
[a434]     write
[a435]   </AttributeValue>
[a436]   <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml: 1.0:action:action-id"
[a437]     DataTyPe="http://www.w3.org/2001/XMLSchema#string"/>
[a438]   </ActionMatch>
[a439] </Action>
[a440] </Actions>
[a441] </Target>

```

```

[a442] <Condition>
[a443] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
[a444] <Apply
[a445] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-
only">
[a446] <SubjectAttributeDesignator
[a447] AttributeId="urn:oasis:names:tc:xacml:2.0:example:
attribute:physician-id"
[a448] DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a449] </Apply>
[a450] <Apply
[a451] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-
only">
[a452] <AttributeSelector RequestContextPath=
[a453] "//xacml-context:Resource/xacml-
context:ResourceContent/md:record/
md:primaryCarePhysician/md:registrationID/text()"
[a454] DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a455] </Apply>
[a456] </Apply>
[a457] </Condition>
[a458] </Rule>
[a459] <Obligations>
[a460] <Obligation
ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
[a461] FulfillOn="Permit">
[a462] <AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:mailto"
[a463] DataType="http://www.w3.org/2001/XMLSchema#string">
[a464] <AttributeSelector RequestContextPath=
[a465] "//md:/record/md:patient/md:patientContact/md:email"
[a466] DataType="http://www.w3.org/2001/XMLSchema#string"/> ;
[a467] </AttributeAssignment>
[a468] <AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:text"
[a469] DataType="http://www.w3.org/2001/XMLSchema#string">
[a470] Your medical record has been accessed by:
[a471] </AttributeAssignment>
[a472] <AttributeAssignment
AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:text"
[a473] DataType="http://www.w3.org/2001/XMLSchema#string">
[a474] <SubjectAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[a475] DataType="http://www.w3.org/2001/XMLSchema#string"/> ;
[a476] </AttributeAssignment>
[a477] </Obligation>
[a478] </Obligations>
[a479] </Policy>

```

[a366] – [a372] <Policy> 单元包括标准命名空间声明以及策略特定参数，如 PolicyId 和 RuleCombiningAlgId。

[a371]策略标识符。该参数允许策略集引用该策略。

[a372]组合规则算法标识用于组合规则赋值的输出算法。

[a373] – [a376]该策略的自由模式描述。

[a379] – [a394]策略目标。该策略目标规定可应用的决定请求。<Policy>中的<Target>单元结构与<Rule>中的 <Target> 单元结构相同。在这种情况下，策略目标是符合命名空间 "urn:example:med:schemas:record"的所有 XML 资源集。

[a395]该<Policy>中仅包括<Rule>单元。在规则头中规定两个参数：RuleId 和 Effect。

[a402] – [a441]规则目标进一步限制策略目标。

[a405] – [a412]<SubjectMatch>单元确定其"urn:oasis:names:tc:xacml:2.0:example:attribute:role"主体属性等于"physician"的主体的规则目标。

[a417] – [a425]<ResourceMatch>单元为匹配 Xpath 表达式"/md:record/md:medical"的资源确定规则目标。

[a430] – [a438]<ActionMatch>单元为其"urn:oasis:names:tc:xacml:1.0:action:action-id"动作属性等于"write"的动作确定规则。

[a442] – [a457]<Condition>单元。对于可应用于决定亲的规则，条件必须赋值为“True”。该条件将"urn:oasis:names:tc:xacml:2.0:example:attribute:physician-id"主体属性的值与在将接入到病历档案中的<registrationId>单元的值进行比较。

[a459] – [a478]<Obligations>单元。职责是必须由与授权决定连同 PEP 来执行的操作集。职责可以与“Permit”或“Deny”授权决定相关联。该单元包含单一的职责。

[a460] – [a477]<Obligation>单元由 ObligationId 属性，必须填写的授权决定值以及属性分配集组成。PDP 不解析属性分配。这是 PEP 的责任。

[a460]ObligationId 属性标识职责。在这种情况下，需要 PEP 发送 E-mail。

[a461]FulfillOn 属性规定该职责必须被填充的授权决定值。在该情况下，允许接入。

[a462] – [a467]第一个参数指示 PEP 将在资源的何处找到 E-mail 地址。

[a468] – [a471]第二个参数包含 E-mail 主体的文字文本。

[a472] – [a476]第三个参数指示 PEP 将在资源的何处找到 E-mail 主体的进一步文本。

II.2.4.4 规则4

规则 4 举例说明“Deny”Effect 值以及不包含<Condition>单元的<Rule>的使用。

```
[a480] <?xml version="1.0" encoding="UTF-8"?>
[a481] <Policy
[a482] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
[a483] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-
os.xsd"
[a484] xmlns:md="http://www.med.example.com/schemas/record.xsd"
[a485] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:4"
[a486] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm: deny-overrides">
[a487] <PolicyDefaults>
[a488] <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
19991116</XPathVersion>
[a489] </PolicyDefaults>
[a490] <Target/>
[a491] <Rule
[a492] RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:4"
[a493] Effect="Deny">
[a494] <Description>
[a495] An Administrator shall not be permitted to read or write
[a496] medical elements of a patient record in the
[a497] http://www.med.example.com/records.xsd namespace.
[a498] </Description>
[a499] <Target>
[a500] <Subjects>
[a501] <Subject>
[a502] <SubjectMatch
[a503] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a504] <AttributeValue
DataTypes="http://www.w3.org/2001/XMLSchema#string">
[a505] administrator
[a506] </AttributeValue>
[a507] <SubjectAttributeDesignator AttributeId=
[a508] "urn:oasis:names:tc:xacml:2.0:example:attribute:role"
[a509] DataTypes="http://www.w3.org/2001/XMLSchema#string"/>
[a510] </SubjectMatch>
```

```

[a511]     </Subject>
[a512]   </Subjects>
[a513] <Resources>
[a514]   <Resource>
[a515]     <ResourceMatch
[a516]       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a517]         <AttributeValue
DataTypes="http://www.w3.org/2001/XMLSchema#string">
[a518]           urn:example:med:schemas:record
[a519]         </AttributeValue>
[a520]         <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
[a521]           DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a522]         </ResourceMatch>
[a523]       <ResourceMatch
[a524]         MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-
match">
[a525]           <AttributeValue
DataTypes="http://www.w3.org/2001/XMLSchema#string">
[a526]             /md:record/md:medical
[a527]           </AttributeValue>
[a528]           <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
[a529]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a530]           </ResourceMatch>
[a531]         </Resource>
[a532]       </Resources>
[a533]     <Actions>
[a534]       <Action>
[a535]         <ActionMatch
[a536]           MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a537]             <AttributeValue
DataTypes="http://www.w3.org/2001/XMLSchema#string">
[a538]               read
[a539]             </AttributeValue>
[a540]             <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[a541]               DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a542]             </ActionMatch>
[a543]           </Action>
[a544]         <Action>
[a545]           <ActionMatch
[a546]             MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a547]               <AttributeValue
DataTypes="http://www.w3.org/2001/XMLSchema#string">
[a548]                 write
[a549]               </AttributeValue>
[a550]               <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[a551]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a552]               </ActionMatch>
[a553]             </Action>
[a554]           </Actions>
[a555]       </Target>
[a556]     </Rule>
[a557] </Policy>

```

[a492] – [a493]<Rule>单元声明。

[a493]规则 Effect。赋值为“True”的每一个规则将该规则作为其值发送。该规则 Effect 是“Deny”，其含义是按照该规则，在它赋值为“True”时，接入必须被拒绝。

[a494] – [a498]规则的自由形态描述。

[a499] – [a555]规则目标。规则目标规定可应用于该规则的决定请求集。

[a502] – [a510]<SubjectMatch>单元为其"urn:oasis:names:tc:xacml:2.0:example:attribute:role"主体属性等于“管理者”的主体制定规则目标。

[a513] – [a532]<Resources>单元包含一个<Resource>单元，该单元（依次）包含两个<ResourceMatch>单元。若由请求上下文标识的资源匹配两个资源匹配准则，目标单元匹配。

[a515] – [a522]第一个<ResourceMatch>单元以其"urn:oasis:names:tc:xacml:2.0:resource:target-namespace"资源属性等于"urn:example:med:schemas:record"的资源为目标规则。

[a523] – [a530]第二个<ResourceMatch>单元将匹配 XPath 表达式 "/md:record/md:medical"的 XML 单元为目标规则。

[a533] – [a554]<Actions>单元包含两个<Action>单元，每一个包含一个<ActionMatch>单元。若在请求上下文中标识的动作匹配任一个动作匹配准则，目标匹配。

[a535] – [a552]<ActionMatch>单元确定其"urn:oasis:names:tc:xacml:1.0:action:action-id"动作属性等同于“阅读”或“写”的动作定位目标。

该规则没有<Condition>单元。

II.2.4.5 示例PolicySet

本部分使用前面描述组合策略过程的先前条款的示例。对档案的病历内容的读接入控制由 II.4.2.3 中描述的四个规则中的每一个来形成。用一般的语言来讲，组合规则是：

- 请求者是患者；或
- 请求者是不满16岁的患者的父母或监护人；或
- 请求者是主治医师同时向患者发送了通知；并且
- 请求者不是管理者。

下列策略集用图表的方式给出了组合的策略。通过应用包括策略 3 同时明确地包括策略 2。

```
[a558] <?xml version="1.0" encoding="UTF-8"?>
[a559] <PolicySet
[a560]   xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
[a561]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-
os.xsd"
[a562]   PolicySetId=
[a563]     "urn:oasis:names:tc:xacml:2.0:example:policysetid:1"
[a564]   PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:
[a565]     policy-combining-algorithm:deny-overrides">
[a566]     <Description>
[a567]       Example policy set.
[a568]     </Description>
[a569]     <Target>
[a570]       <Resources>
[a571]         <Resource>
[a572]           <ResourceMatch
[a573]             MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a574]               <AttributeValue
[DataType="http://www.w3.org/2001/XMLSchema#string">
[a575]                 urn:example:med:schema:records
[a576]               </AttributeValue>
[a577]             <ResourceAttributeDesignator AttributeId=
[a578]               "urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
[a579]               DataType="http://www.w3.org/2001/XMLSchema#string"/>
[a580]             </ResourceMatch>
[a581]           </Resource>
[a582]         </Resources>
[a583]       </Target>
[a584]     <PolicyIdReference>
[a585]       urn:oasis:names:tc:xacml:2.0:example:policyid:3
[a586]     </PolicyIdReference>
[a587]     <Policy
[a588]       PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:2"
```

```

[a589] RuleCombiningAlgId=
[a590] "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-
overrides">
[a591] <Target/>
[a592] <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1"
[a593] Effect="Permit">
[a594] </Rule>
[a595] <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:2"
[a596] Effect="Permit">
[a597] </Rule>
[a598] <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:4"
[a599] Effect="Deny">
[a600] </Rule>
[a601] </Policy>
[a602] </PolicySet>

```

[a559] – [a565] <PolicySet> 单元声明。包括标准 XML 命名空间声明。

[a562] PolicySetId 属性用于标识在另一个策略集中可能包含的该策略集。

[a564] 策略组合算法标识符。在计算授权决定时，根据特定的策略组合算法组合该策略集中的策略和策略集。

[a566] – [a568] 该策略集的自由形式描述。

[a569] – [a583] 策略集 <Target> 单元规定可应用于该 <PolicySet> 单元的决定请求集。

[a584] PolicyIdReference 通过 id 包括策略。

[a588] Policy 2 被明确地包含在该策略集中。为清楚起见，在 Policy 2 的规则中被省略。

附录三

高阶包函数示例描述

III.1 高阶包函数示例

本附录描述包进行操作的 XACML 中的函数，以便于可以将函数通常应用于包。

为举例的目的，将称为 Haskell (见 [Haskell]) 的多用途函数语言规定这些函数的语义。尽管英文描述已经足够了，语义的适当描述是有帮助的。

很快总结一下，在下面的 Haskell 符号中，函数定义采用可应用于称为列表的结构模式的形式。符号 “[]” 表示空表，然而，表达式 “(x:xs)” 不匹配非空表的变量，此处 “x” 表示列表中的第一个单元，同时 “xs” 是列表的其余部分，该列表可能是空列表。我们使用列表的 Haskell 注释，该列表是有序的单元集合，模型化值的 XACML 包。

取类型布尔的一系列值的常用函数 “urn:oasis:names:tc:xacml:1.0:function:and” 的简单 Haskell 定义规定如下：

```

and:: [Bool]    -> Bool

and []          = True

and (x:xs)     = x && (and xs)

```

由 “::” 表示的第一个定义行正式地描述函数的数据类型，该函数采用由 “[Bool]” 表示的一系列布尔值，返回用 “Bool” 表示的一个布尔值。第二个定义行是一个说明应用到空列表的函数 “与” 是 “True” 的一个条款。第三个定义行是陈述对于非空列表，第一个单元是数据类型 Bool 的值 “x”，应采用中缀符号 “&&” 表示的逻辑关联函数，将应用于 x 的函数 “and” 与循环应用于该列表其余部分的函数 “and” 组合起来。当然，在只有在应用于其的列表是空或者该列表的每一个单元是 “True” 的情况下，“and” 函数的应用是 “True”。例如，下列 Haskell 表达式 (and [])，(and [True])，(and [True,True])，(and [True,True,False]) 分别赋值为 “True”、“True”、“True” 和 “False”。

1) urn:oasis:names:tc:xacml:1.0:function:any-of

在Haskell中, 该操作的语义如下:

```
any_of :: ( a -> b -> Bool )    -> a -> [b] -> Bool
```

```
any_of f a []                    = False
```

```
any_of f a (x:xs)                = (f a x) || (any_of f a xs)
```

在上面的符号中, "f"是将要应用的函数, "a" 是初始值, 同时"(x:xs)"表示列表的第一个单元表示为 "x", 列表的其余部分表示为 "xs".

例如, 下面的表达式应返回 "True":

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
  <Function
    FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
    equal"/>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
    Paul
  </AttributeValue>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      John
    </AttributeValue>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      Paul
    </AttributeValue>
    <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#string">George
    </AttributeValue>
    <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#string">
      Ringo
    </AttributeValue>
  </Apply>
</Apply>
```

因为按照该函数, 第一个变量等于包的至少一个单元, 该表达式是 "True".

2) urn:oasis:names:tc:xacml:1.0:function:all-of

在Haskell中, 该操作的语义如下:

```
all_of :: ( a -> b -> Bool )    -> a -> [b] -> Bool
```

```
all_of f a []                    = True
```

```
all_of f a (x:xs)                = (f a x) && (all_of f a xs)
```

在上面的符号中, "f"是要应用的函数, "a"是初始值, 同时"(x:xs)"表示, 列表的第一个单元为 "x", 列表的其余部分为 "xs".

例如, 下列表达式应赋值为 "True".

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of">
  <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
  greater"/>
  <AttributeValue
    DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
    <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#integer">9</AttributeValue>
    <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
    <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
    <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
  </Apply>
</Apply>
```


因为第一个参数（10）比该包（9,3,4和2）中的所有单元均大,所以该表达式是“True”。

3) urn:oasis:names:tc:xacml:1.0:function:any-of-any

在Haskell中, 利用上面规定的“任一”函数, "any_of_any"函数如下:

```
any_of_any :: ( a -> b -> Bool )      -> [a]-> [b] -> Bool
any_of_any f []                        ys                = False
any_of_any f (x:xs)                    ys                = (any_of f x ys) || (any_of_any f xs ys)
```

在上面的符号中, "f"是将应用的函数, 同时"(x:xs)"表示列表中的第一个单元是“x”, 列表中的其他单元为“xs”。

例如, 下列表达式应赋值为“True”:

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-any">
  <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal"/>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Mary</AttributeValue>
  </Apply>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string"> John</AttributeValue>
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string"> Paul</AttributeValue>
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string"> George</AttributeValue>
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string"> Ringo</AttributeValue>
  </Apply>
</Apply>
```

因为第一个包的至少一个命名为“Ringo”的单元等于第二个包的至少一个单元, 该表达式是“True”。

4) urn:oasis:names:tc:xacml:1.0:function:all-of-any

在Haskell中, 在上面Haskell中规定的"any_of"函数, "all_of_any"函数的语义如下:

```
all_of_any :: ( a -> b -> Bool )      -> [a]-> [b] -> Bool
all_of_any f []                        ys                = True
all_of_any f (x:xs)                    ys                = (any_of f x ys) && (all_of_any f xs ys)
```

在上面的符号中, "f"是将应用的函数, 同时"(x:xs)"表示列表中的第一个单元是“x”, 列表中的其他单元为“xs”。

例如, 下列表达式应赋值为“True”:

```
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-any">
  <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
greater"/>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">20</AttributeValue>
  </Apply>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
```

```

        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"> 3</AttributeValue>
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"> 5</AttributeValue>
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"> 19</AttributeValue>
    </Apply>
</Apply>

```

因为第一个包中的每一个单元比第二个包的至少一个单元大，所以该表达式是“True”。

5) urn:oasis:names:tc:xacml:1.0:function:any-of-any

在Haskell中，利用上面规定的“all_of”函数，“any_of_any”函数的语义如下：

```

any_of_all :: (a -> b -> Bool)    -> [a]-> [b] -> Bool
any_of_all f []          ys        = False
any_of_all f (x:xs)     ys        = (all_of f x ys) || (any_of_all f xs ys)

```

在上面的符号中，“f”是将应用的函数，同时“(x:xs)”表示列表中的第一个单元是“x”，列表中的其他单元为“xs”。

例如，下列表达式应赋值为“True”：

```

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-all">
  <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
greater"/>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"> 3</AttributeValue>
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"> 5</AttributeValue>
  </Apply>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"> 1</AttributeValue>
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"> 2</AttributeValue>
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"> 3</AttributeValue>
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"> 4</AttributeValue>
  </Apply>
</Apply>

```

因为对于第二个包中的所有值，在第一个包中有一个比较大的值，所以该表达式是“True”。

6) urn:oasis:names:tc:xacml:1.0:function:all-of-all

在Haskell中，利用上面规定的“all_of”函数，“any_of_any”函数的语义如下：

```

any_of_any :: (a -> b -> Bool)    -> [a]-> [b] -> Bool
any_of_any f []          ys        = False
any_of_any f (x:xs)     ys        = (any_of f x ys) && (any_of_any f xs ys)

```

在上面的符号中，“f”是将应用的函数，同时“(x:xs)”表示列表中的第一个单元是“x”，列表中的其他单元为“xs”。

例如，下列表达式应赋值为“True”：

```

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-all">
  <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
greater"/>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"> 6</AttributeValue>
    <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#integer"> 5</AttributeValue>
  </Apply>
</Apply>

```

```

    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
      <AttributeValue
DataTyPe="http://www.w3.org/2001/XMLSchema#integer"> 1</AttributeValue>
      <AttributeValue
DataTyPe="http://www.w3.org/2001/XMLSchema#integer"> 2</AttributeValue>
      <AttributeValue
DataTyPe="http://www.w3.org/2001/XMLSchema#integer"> 3</AttributeValue>
      <AttributeValue
DataTyPe="http://www.w3.org/2001/XMLSchema#integer"> 4</AttributeValue>
    </Apply>
  </Apply>

```

因为对于第一个包中的所有单元，“5”和“6”，均比第二个包的所有整数值“1”，“2”，“3”，“4”大，所以该表达式是“True”。

7) urn:oasis:names:tc:xacml:1.0:function:map

在Haskell中，该函数规定如下：

```
map:: (a -> b) -> [a] -> [b]
```

```
map f []      = []
```

```
map f (x:xs) = (f x):(map f xs)
```

在上面的符号中，“f”是将应用的函数，同时“(x:xs)”表示列表中的第一个单元是“x”，列表中的其他单元为“xs”。

例如，下列表达式应赋值为“True”：

```

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:map">
  <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
normalize-to-lower-case">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
      <AttributeValue
DataTyPe="http://www.w3.org/2001/XMLSchema#string"> Hello</AttributeValue>
      <AttributeValue
DataTyPe="http://www.w3.org/2001/XMLSchema#string"> World!</AttributeValue>
    </Apply>
  </Apply>

```

赋值包含有“hello”和“world!”的包。

参考资料

- [Haskell] THOMPSON (S.): Haskell: The Craft of Functional Programming (2nd Edition), *Addison Wesley*, ISBN 0-201-34275-8, 1996.
- [IEEE 754] IEEE 754-1985, *Binary Floating-Point Arithmetic*, ISBN 1-5593-7653-8, IEEE Product No. SH10116-TBR.
- [RBAC] ANSI INCITS 359-2004, *Information technology – Role Based Access Control*, <http://csrc.nist.gov/rbac/>.

ITU-T 系列建议书

A系列	ITU-T工作的组织
D系列	一般资费原则
E系列	综合网络运行、电话业务、业务运行和人为因素
F系列	非话电信业务
G系列	传输系统和媒质、数字系统和网络
H系列	视听及多媒体系统
I系列	综合业务数字网
J系列	有线网络和电视、声音节目及其它多媒体信号的传输
K系列	干扰的防护
L系列	电缆和外部设备其它组件的结构、安装和保护
M系列	电信管理，包括TMN和网络维护
N系列	维护：国际声音节目和电视传输电路
O系列	测量设备的技术规范
P系列	电话传输质量、电话设施及本地线路网络
Q系列	交换和信令
R系列	电报传输
S系列	电报业务终端设备
T系列	远程信息处理业务的终端设备
U系列	电报交换
V系列	电话网上的数据通信
X系列	数据网、开放系统通信和安全性
Y系列	全球信息基础设施、互联网协议问题和下一代网络
Z系列	用于电信系统的语言和一般软件问题