XACML v3.0 Related and Nested Entities Profile Version 1.0

Committee Specification 03

30 January 2024

This stage:
https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/cs03/xacml-3.0-related-entities-v1.0-cs03.docx (Authoritative)
https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/cs03/xacml-3.0-related-entities-v1.0-cs03.html
https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/cs03/xacml-3.0-related-entities-v1.0-cs03.pdf

Previous stage:
https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/csd03/xacml-3.0-related-entities-v1.0-csd03.docx (Authoritative)
https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/csd03/xacml-3.0-related-entities-v1.0-csd03.html
https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/csd03/xacml-3.0-related-entities-v1.0-csd03.pdf

Latest stage:
https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/xacml-3.0-related-entities-v1.0.docx (Authoritative)
https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/xacml-3.0-related-entities-v1.0.html
https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/xacml-3.0-related-entities-v1.0.pdf

Technical Committee:
OASIS eXtensible Access Control Markup Language (XACML) TC

Chairs:
Hal Lockhart (harold.w.lochhart@gmail.com), Individual
Bill Parducci (bill@parducci.net), Individual

Editor:
Steven Legg (steven.legg@viewds.com), ViewDS Identity Solutions

Additional artifacts:
This prose specification is one component of a Work Product that also includes:

- XML schema: https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/cs03/schemas/

Related work:
This specification is related to:

- *eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01.* Edited by Erik Rissanen. 12 July 2017. OASIS Standard incorporating Approved Errata. http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-complete.html. Latest version: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html.

Declared XML namespaces:

- urn:oasis:names:tc:xacml:3.0:core:schema:wd-17

Abstract:
It is not unusual for access control policy to be dependent on attributes that are not naturally properties of the access subject or resource, but rather are properties of ***entities*** that are *related* to the access subject or resource. This profile defines the means to reference such attributes from within XACML policies for processing by a policy decision point.

Status:
This document was last revised or approved by the OASIS eXtensible Access Control Markup Language (XACML) TC on the above date. The level of approval is also listed above. Check the "Latest stage" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/xacml/.

This specification is provided under the RF on Limited Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/xacml/ipr.php).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:
When referencing this specification, the following citation format should be used:

[xacml-3.0-nested-ent-v1.0]

XACML v3.0 Related and Nested Entities Profile Version 1.0. Edited by Steven Legg. 30 January 2024. Committee Specification 03. https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/cs03/xacml-3.0-related-entities-v1.0-cs03.html. Latest stage: https://docs.oasis-open.org/xacml/xacml-3.0-related-entities/v1.0/xacml-3.0-related-entities-v1.0.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website: [https://www.oasis-open.org/policies-guidelines/ipr/].

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. OASIS AND ITS MEMBERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THIS DOCUMENT OR ANY PART THEREOF.

As stated in the OASIS IPR Policy, the following three paragraphs in brackets apply to OASIS Standards Final Deliverable documents (Committee Specifications, OASIS Standards, or Approved Errata).

[OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this deliverable.]

[OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this OASIS Standards Final Deliverable. OASIS may include such claims on its website, but disclaims any obligation to do so.]

[OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this OASIS Standards Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Standards Final Deliverable, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.]

The name "OASIS" is a trademark of OASIS, the owner and developer of this document, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, documents, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark/ for above guidance.

Table of Contents

1 Introduction

1.1 Overview

{Non-normative}

The eXtensible Access Control Markup Language (XACML) **[XACML3]** defines categories of attributes that describe ***entities*** of relevance to access control decisions. XACML rules, policies and policy sets contain assertions over the attributes of these ***entities*** that must be evaluated to arrive at an access decision. Principal among the various predefined ***entities*** are the ***entity*** that is requesting access, i.e., the access subject, and the ***entity*** being accessed, i.e., the resource. However, it is not unusual for access decisions to be dependent on attributes of ***entities*** that are associated with the access subject or resource. For example, attributes of an organization that employs the access subject, or attributes of a licensing agreement that covers the terms of use of a resource.

This profile defines two ways of representing these associated entities in the request context - ***related entities*** and ***nested entities*** - and defines additional mechanisms to access and traverse these ***entities***.

Nested entities are represented by a new data-type (the ***entity data-type***) which carries all the attributes of the ***entity*** as its value. A ***nested entity*** appears in the request context as a sub-element of the containing ***entity***.

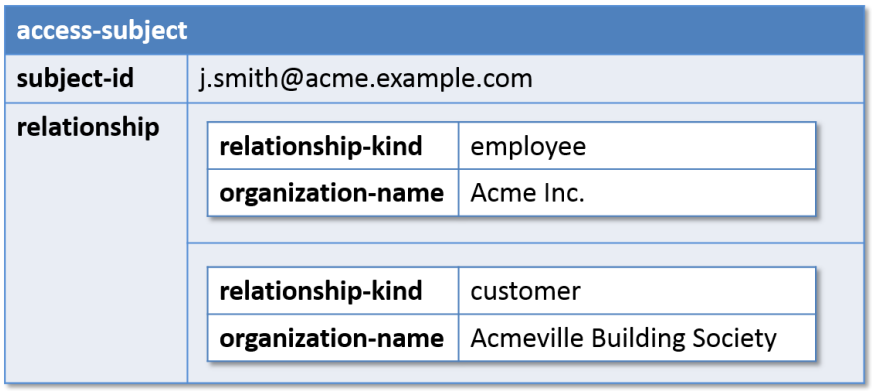

Figure 1 - An access-subject with nested entities

The ***entity data-type*** also provides a way for XACML to represent and process values of structured data that does not depend on using XML documents and XPath expressions.

Related entities appear alongside the predefined ***entities*** in the request context and have the same structure. The `Category` XML attribute **[XML]** of a ***related entity*** holds a distinct, system-defined URI that may be used in XACML attribute values to refer to the ***related entity*** from another ***entity***.

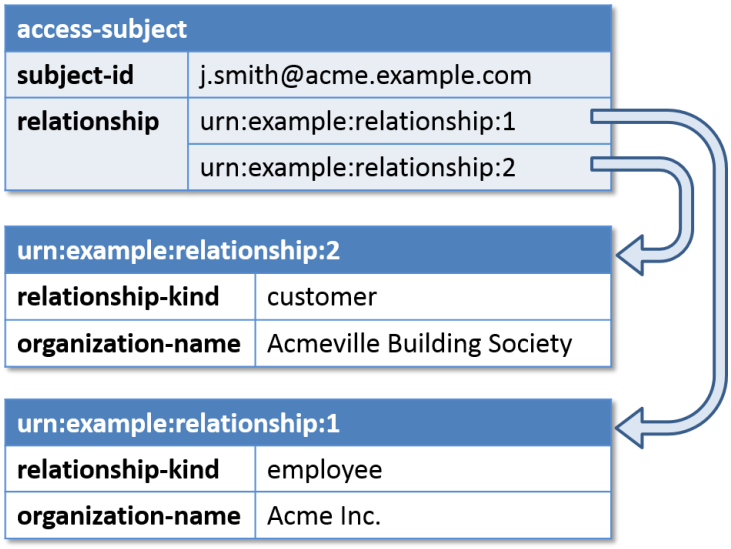

Figure 2 - An access-subject with references to related entities

This profile defines the attribute-designator and the attribute-selector functions (function variants of the `<AttributeDesignator>` and `<AttributeSelector>` elements in XACML 3.0) to enable the extraction of attributes from ***nested entities*** and ***related entities***. A policy author does not need any knowledge of a ***related entity's*** URI in order to extract attributes from it using these functions.

This profile also defines ***quantified expressions***, which are a generalization of the higher-order bag functions of XACML 3.0 for manipulating bags of values without the need for an explicit looping construct. By declaring an explicit ***quantified variable***, the ***quantified expressions*** may be nested without ambiguity. The quantified expressions allow non-trivial expressions to be applied to the members of a bag, including members that are associated ***entities***.

1.2 Glossary

Domain

The domain of discourse of a ***quantified variable***, which is the set of values a ***quantified variable*** is allowed to take during the evaluation of a ***quantified expression***. The ***domain*** is represented by a bag of values.

Entity

A collection of XACML attributes and/or XML content describing the properties of a thing that exists, though possibly in a conceptual rather than physical sense. A person, organization, device, database record, employment contract, agreement or regulation are each an example of an ***entity***. The predefined attribute categories of XACML such as access subject, resource and action are also examples of ***entities***.

Entity data-type

An XACML data-type for representing ***entities*** as attribute values.

Iterant expression

An XACML expression to be separately evaluated for each value of a ***quantified variable***.

Nested entity

An ***entity*** that appears as a value of an attribute of another ***entity***.

Quantified expression

An XACML expression that nominates a ***quantified variable***, a ***domain*** and an ***iterant expression***. A ***quantified expression*** is evaluated by combining the results of evaluating the

iterant expression multiple times, each time binding the ***quantified variable*** to a different value from the ***domain***.

Quantified variable

A variable that ranges over the values in a ***domain*** and is bound to those values by a ***quantified expression.***

Related entity

An ***entity*** in the request context that is separate from every other ***entity*** and is referenced by a URI.

1.3 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in BCP 14 **[RFC2119]** and **[RFC8174]** when, and only when, they appear in all capitals, as shown here.

XML Schema **[XSD1][XSD2]** fragments in this specification are non-normative and assume the following two namespace declarations:

- `xmlns:xs="http://www.w3.org/2001/XMLSchema"`
- `xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"`

The replacement text for the XML entity reference “`&xacml;`” used in examples is `"urn:oasis:names:tc:xacml:"`.

1.4 Normative References

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[RFC3986] Berners-Lee, T., Fielding, R. and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax”, RFC 3986, January 2005. http://www.ietf.org/rfc/rfc3986.txt.

[RFC4627] Crockford, D., “The application/json Media Type for JavaScript Object Notation (JSON)”, RFC 4627, July 2006. http://www.ietf.org/rfc/rfc4627.txt.

[RFC8174] Leiba, B., “Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words”, BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, http://www.rfc-editor.org/info/rfc8174.

[XACML3] *eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01.* Edited by Erik Rissanen. 12 July 2017. OASIS Standard incorporating Approved Errata. http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-complete.html. Latest version: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html.

[XACMLJSON] *JSON Profile of XACML 3.0 Version 1.1.* Edited by David Brossard and Steven Legg. 20 June 2019. OASIS Standard. https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/os/xacml-json-http-v1.1-os.html. Latest version: https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/xacml-json-http-v1.1.html.

[XF] *XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition)*, W3C Recommendation 14 December 2010. Available at: http://www.w3.org/TR/2010/REC-xpath-functions-20101214/.

[XML] Extensible Markup Language (XML) 1.0 (Fifth Edition), T. Bray, J. Paoli, M. Sperberg-McQueen, E. Maler, F. Yergeau, Editors, W3C Recommendation, November 26, 2008, http://www.w3.org/TR/2008/REC-xml-20081126/. Latest version available at http://www.w3.org/TR/xml/.

[XPath] *XML Path Language (XPath), Version 1.0*, W3C Recommendation 16 November 1999. Available at: http://www.w3.org/TR/xpath.

[XSD1] *XML Schema Part 1: Structures Second Edition*, W3C Recommendation 28 October 2004. Available at: http://www.w3.org/TR/xmlschema-1/.

[XSD2] *XML Schema Part 2: Datatypes Second Edition*, W3C Recommendation 28 October 2004. Available at: http://www.w3.org/TR/xmlschema-2/.

1.5 Non-Normative References

[RFC6963] Saint-Andre, P., "A Uniform Resource Name (URN) Namespace for Examples", RFC 6963, May 2013. http://www.ietf.org/rfc/rfc6963.txt.

[SAML] *SAML 2.0 Profile of XACML, Version 2.0*. 19 August 2014. OASIS Committee Specification 02. Latest version: http://docs.oasis-open.org/xacml/xacml-saml-profile/v2.0/xacml-saml-profile-v2.0.html.

[ADMIN] *XACML v3.0 Administration and Delegation Profile Version 1.0*. 13 November 2014. OASIS Committee Specification Draft 04 / Public Review Draft 02. Latest version: http://docs.oasis-open.org/xacml/3.0/administration/v1.0/xacml-3.0-administration-v1.0.html.

2 Background

{Non-normative}

There are a number of approaches for dealing with entities associated with the access subject, resource or other predefined entities.

An obvious approach is to define additional categories for these associated ***entities***. However, this becomes problematic when there can be more than one instance of a particular kind of ***entity***. For example, if the access subject is employed by more than one organization. Sufficient additional organization categories must be defined to accommodate the maximum possible number of associated organizations (e.g., organization-1, organization-2, …, organization-n), but each additional organization category increases the complexity of XACML policies since the policy writer must account for the worst case, making the same assertions over each organization category.

Another solution is to have the policy enforcement point (PEP) or policy information point (PIP) make the attributes of associated ***entities*** appear as attributes of the access subject or resource in a process referred to as “flattening”. Flattening of associated ***entities*** is adequate in simple cases where the access subject or resource has a relationship with at most one instance of each kind of associated ***entity*** (for example, the access subject can be employed by only one organization), or where the associations between attribute values originating from the *same* associated ***entity*** are not important (for example, if it is not necessary to know whether a particular organization employing the access subject is both not-for-profit *and* registered in a particular country).

Where it is necessary to apply some test over attributes of the same associated ***entity***, one possible approach is to have the PEP or PIP generate a derived attribute that holds the result of an expression evaluated over the attributes of an associated ***entity*** (for example, a Boolean attribute of the access subject whose value indicates whether the access subject is employed by a not-for-profit organization registered in the USA). The disadvantage of this approach is that part of the access control logic resides in the implementation or configuration of the PEP or PIP (instead of in XACML policies) where it is not readily visible to policy writers, not easily changed by policy writers and not easily transportable to other XACML deployments.

This profile offers a different approach that avoids the drawbacks of the preceding solutions. It establishes conventions for representing associated ***entities*** and the links between them and defines the means to evaluate XACML expressions over the attributes of associated ***entities*** that are discovered at the time of policy evaluation.

3 Nested and Related Entities

Two representations of associated ***entities*** are defined by this profile: ***related entities*** and ***nested entities***.

Related entities are separate ***entities*** in the request context that are identified by arbitrary URIs **[RFC3986]** using their `Category` XML attribute. These URIs are system defined, distinct from the predefined category URIs and not necessarily known to policy writers. An ***entity*** MAY reference a ***related entity*** through an XACML attribute value of the `http://www.w3.org/2001/XMLSchema#anyURI` data-type that holds the URI of the ***related entity***. This profile places no restrictions on the form of the URI except that it MUST be unique to each ***related entity*** in the request context and references must be consistent. The order of ***related entities*** in the request context carries no significance.

A ***nested entity*** has the same composition as a ***related entity*** except that it lacks a `Category` or `Id` XML attribute and appears as a value of an attribute of another ***entity*** using the ***entity data-type*** defined in Section 4.

Nested entities are suited to cases where ***entities*** are associated hierarchically. ***Related entities*** are suited to cases where there are multiple kinds of reference between the same pair of ***entities*** or where the chains of references can form cycles or link the access subject to the resource (such cases can be handled by nesting, but not without duplication of ***entities*** in the request context). There is no prohibition on using ***related entities*** for some kinds of ***entities*** and nesting for the remaining ***entities***.

4 The Entity Data-type

The ***entity data-type*** is used to represent an ***entity*** nested within another ***entity***. It is identified by the URI `urn:oasis:names:tc:xacml:3.0:data-type:entity` . The ***entity data-type*** MAY be used in places where XACML **[XACML3]** requires a primitive data-type. In particular, this means that it is possible to have a bag of ***entities***.

The valid XML representation for a value of this data-type is an optional `<Content>` child element followed by zero or more `<Attribute>` child elements.

XACML attribute values in the XML representation are validated according to the `AttributeValueType` XML Schema type. An XML Schema type definition for the content of a value of the ***entity data-type*** would not be used and is therefore not required. However, to aid understanding, the content of an `<AttributeValue>` element for a value of the ***entity data-type*** would be expected to conform to the following `EntityType` XML Schema type:

```
<xs:complexType name="EntityType">
  <xs:sequence>
    <xs:element ref="xacml:Content" minOccurs="0"/>
    <xs:element ref="xacml:Attribute" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The JSON representation **[RFC4627]** of an attribute value of the ***entity data-type*** follows the representation of a Category object **[XACMLJSON]** except that name/value pairs for “Category” and “Id” SHALL NOT appear.

The JSON shorthand type code **[XACMLJSON]** for the ***entity data-type*** is “entity”. This data-type MUST always be explicitly given in the JSON representation; it cannot be inferred from an attribute value.

The result of successfully evaluating the XPath expression **[XPath]** in the `Path` XML attribute of an `<AttributeSelector>` element **[XACML3]** is a node-set that is to be converted to a bag of XACML attribute values of the data-type nominated by the `DataType` XML attribute of the `<AttributeSelector>` element. The conversion to values of the ***entity data-type*** is defined here. If any node in the node-set is not an element node, then the attribute selector SHALL return “Indeterminate” with the status code `urn:oasis:names:tc:xacml:1.0:status:syntax-error`; otherwise, an attribute value of the ***entity data-type*** SHALL be generated for each node in the node-set. In terms of the XML representation, each such attribute value SHALL have a `<Content>` child element and SHALL NOT have any `<Attribute>` child elements. The child element of the `<Content>` element SHALL be a copy of the element corresponding to the node, along with its entire content, plus whatever namespace declarations from ancestor elements as are required to define namespace prefixes used in the content. Namespace declarations from ancestor elements that are not visibly used in the content MAY be added.

In terms of the JSON representation, each value SHALL have a “Content” name/value pair and SHALL NOT have an “Attribute” name/value pair. The value of the “Content” name/value pair SHALL be the Base64 encoding or escaped XML string representation (see **[XACMLJSON]**) of the child XML element of the `<Content>` element that would be constructed for the XML representation.

4.1 Examples of Entity Values

{Non-normative}

Figure 3 shows an attribute value of the ***entity data-type*** in both XML and JSON. This value also contains a ***nested entity***.

```
XML:
```

```
  <Attribute xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    IncludeInResult="false"
    AttributeId="urn:example:xacml:attribute:relationship">
    <AttributeValue DataType="urn:oasis:names:tc:xacml:3.0:data-type:entity">
      <Attribute IncludeInResult="false"
        AttributeId="urn:example:xacml:attribute:relationship-kind">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
           >employee</AttributeValue>
      </Attribute>
      <Attribute IncludeInResult="false"
        AttributeId="urn:example:xacml:attribute:start-date">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#date"
           >2013-09-01</AttributeValue>
      </Attribute>
      <Attribute IncludeInResult="false"
        AttributeId="urn:example:xacml:attribute:organization">
        <AttributeValue
          DataType="urn:oasis:names:tc:xacml:3.0:data-type:entity">
          <Attribute IncludeInResult="false"
            AttributeId="urn:example:xacml:attribute:organization-name">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
              >Acme Inc.</AttributeValue>
          </Attribute>
          <Attribute IncludeInResult="false"
            AttributeId="urn:example:xacml:attribute:organization-type">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
              >commercial</AttributeValue>
          </Attribute>
        </AttributeValue>
      </Attribute>
    </AttributeValue>
  </Attribute>

JSON:

 {
   "AttributeId":"urn:example:xacml:attribute:relationship",
   "DataType":"entity",
   "Value":[{
      "Attribute":[{
        "AttributeId":"urn:example:xacml:attribute:relationship-kind",
        "DataType":"string",
        "Value":["employee"]
      },{
        "AttributeId":"urn:example:xacml:attribute:start-date",
        "DataType":"date",
        "Value":["2013-09-01"]
      },{
        "AttributeId":"urn:example:xacml:attribute:organization",
        "DataType":"entity",
        "Value":[{
          "Attribute":[{
            "AttributeId":"urn:example:xacml:attribute:organization-name",
            "DataType":"string",
            "Value":"Acme Inc."
          },{
            "AttributeId":"urn:example:xacml:attribute:organization-type",
            "DataType":"string",
            "Value":"commercial"
          }]
        }]
```



```
      }]
    }]
  }
```

Figure 3 - An entity in XML and JSON

Figure 4 also shows an attribute value of the ***entity data-type*** in both its XML representation and its equivalent JSON representation. It captures similar information to Figure 3 using a `<Content>` element.

```
XML:

  <Attribute xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    IncludeInResult="false"
    AttributeId="urn:example:xacml:attribute:relationship">
    <AttributeValue DataType="urn:oasis:names:tc:xacml:3.0:data-type:entity">
      <Content>
        <ex:Relationship xmlns:ex="urn:example:xacml:ns:relationship">
          <ex:RelationshipKind>employee</ex:RelationshipKind>
          <ex:StartDate>2013-09-01</ex:StartDate>
          <ex:Organization>
            <ex:OrganizationName>Acme Inc.</ex:OrganizationName>
            <ex:OrganizationType>commercial</ex:OrganizationType>
          </ex:Organization>
        </ex:Relationship>
      </Content>
    </AttributeValue>
  </Attribute>

JSON:

  {
    "AttributeId":"urn:example:xacml:attribute:relationship",
    "DataType":"entity",
    "Value":[{
      "Content":"<?xml version=\"1.0\"?>\n<ex:Relationship xmlns:ex=\"urn:exam
ple:xacml:ns:relationship\">\n          <ex:RelationshipKind>employee</ex:Rela
tionshipKind>\n          <ex:StartDate>2013-09-
01</ex:StartDate>\n          <ex:Organization>\n            <ex:OrganizationNa
me>Acme Inc.</ex:OrganizationName>\n            <ex:OrganizationType>commercia
l</ex:OrganizationType>\n          </ex:Organization>\n        </ex:Relationsh
ip>"
    }]
  }
```

Figure 4 - An entity with XML content

5 Quantified Expressions

There are some common requirements for the ***quantified expressions*** defined in this profile.

The ***quantified expressions*** are additional elements in the substitution group of the `<Expression>` element and all have the same XML Schema type: `QuantifiedExpressionType`.

```
<xs:complexType name="QuantifiedExpressionType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:sequence>
        <xs:element ref="xacml:Expression"/>
        <xs:element ref="xacml:Expression"/>
      </xs:sequence>
      <xs:attribute name="VariableId" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The first child expression is called the ***domain*** and the second child expression is called the ***iterant expression***. The `VariableId` XML attribute names the ***quantified variable*** that will be used by the ***quantified expression***. The ***quantified variable*** does not have a corresponding `<VariableDefinition>`. The ***domain*** SHALL be an expression that evaluates to a bag of values of the same data-type, or "Indeterminate" in the case of an error. The bag MAY be empty. The ***iterant expression*** SHALL be an expression that evaluates to a single value (for all but one kind of ***quantified expression*** the data-type of that value is `http://www.w3.org/2001/XMLSchema#boolean`).

A ***quantified expression*** SHALL NOT use the same `VariableId` as a `<VariableDefinition>` of the enclosing `<Policy>` element. ***Quantified expressions*** MAY be nested in either or both of the ***domain*** and the ***iterant expression***. A nested ***quantified expression*** SHALL NOT use the same `VariableId` as an enclosing ***quantified expression***.

The evaluation of a ***quantified expression*** begins with the evaluation of the ***domain*** expression. The ***iterant expression*** is then evaluated once for each value from the ***domain*** with the ***quantified variable*** set to that value, except that evaluation of the ***quantified expression*** may terminate before considering all the values from the ***domain*** if the final result of the ***quantified expression*** has already been determined. The conditions for early termination are specified for each kind of ***quantified expression***. The effect of the result of the ***iterant expression*** on the overall result of the ***quantified expression*** depends on the kind of ***quantified expression***. Bags of values are unordered so there is no constraint on the order in which an implementation chooses to consider values from the ***domain***.

The value of the ***quantified variable***, i.e., the particular value from the ***domain***, MAY be referenced one or more times from within the ***iterant expression*** using a `<VariableReference>` element. It is not an error if the ***quantified variable*** is not referenced at all. The ***quantified variable*** SHALL NOT be referenced from within the ***domain*** of the ***quantified expression*** to which it belongs. A ***quantified variable*** MAY be referenced from within the ***domain*** of another ***quantified expression*** nested within the ***iterant expression***.

5.1 ForAny Expression

The ForAny ***quantified expression*** tests whether any value of a bag satisfies an ***iterant expression***. It is represented by the `<ForAny>` element, which is of the `QuantifiedExpressionType` complex type.

```
<xs:element name="ForAny" type="xacml:QuantifiedExpressionType"
  substitutionGroup="xacml:Expression"/>
```

The ***iterant expression*** of a ForAny expression SHALL be an expression that evaluates to a value of the `http://www.w3.org/2001/XMLSchema#boolean` data-type.

The result of a ForAny expression SHALL be a value of data-type `http://www.w3.org/2001/XMLSchema#boolean` or “Indeterminate”.

The ForAny expression evaluates to “true” if the ***iterant expression*** evaluates to “true” for any value from the ***domain***; otherwise, the expression evaluates to “Indeterminate” if the ***iterant expression*** evaluates to “Indeterminate” for any value from the ***domain***; otherwise, the expression evaluates to “false”. Note that the ForAny expression evaluates to "false" if the ***domain*** is an empty bag. Evaluation of the expression MAY terminate whenever the ***iterant expression*** evaluates to “true”.

5.2 ForAll Expression

The ForAll ***quantified expression*** tests whether all values of a bag satisfy an ***iterant expression***. It is represented by the `<ForAll>` element, which is of the `QuantifiedExpressionType` complex type.

```
<xs:element name="ForAll" type="xacml:QuantifiedExpressionType"
  substitutionGroup="xacml:Expression"/>
```

The ***iterant expression*** of a ForAll expression SHALL be an expression that evaluates to a value of the `http://www.w3.org/2001/XMLSchema#boolean` data-type.

The result of a ForAll expression SHALL be a value of data-type `http://www.w3.org/2001/XMLSchema#boolean` or “Indeterminate”.

The ForAll expression evaluates to “false” if the ***iterant expression*** evaluates to “false” for any value from the ***domain***; otherwise, the expression evaluates to “Indeterminate” if the ***iterant expression*** evaluates to “Indeterminate” for any value from the ***domain***; otherwise, the expression evaluates to “true”. Note that the ForAll expression evaluates to "true" if the ***domain*** is an empty bag. Evaluation of the expression MAY terminate whenever the ***iterant expression*** evaluates to “false”.

5.3 Map Expression

The Map ***quantified expression*** converts a bag of values to another bag of values. It is represented by the `<Map>` element, which is of the `QuantifiedExpressionType` complex type.

```
<xs:element name="Map" type="xacml:QuantifiedExpressionType"
  substitutionGroup="xacml:Expression"/>
```

The ***iterant expression*** of a Map expression SHALL be an expression that evaluates to a single value (not necessarily of the same data-type as the values in the ***domain***).

The result of the Map expression SHALL be a bag of values of the same data-type as the ***iterant expression*** or “Indeterminate”.

If the ***iterant expression*** evaluates to “Indeterminate” for any value from the ***domain***, then the result of the Map expression is “Indeterminate”; otherwise, the result bag contains the values resulting from the evaluation of the ***iterant expression*** for each value from the ***domain***. The Map expression evaluates to an empty bag if the ***domain*** is an empty bag. Evaluation of the Map expression MAY terminate whenever the ***iterant expression*** evaluates to “Indeterminate”.

5.4 Select Expression

The Select ***quantified expression*** returns a bag containing the values from the ***domain*** that satisfy the ***iterant expression***. That is, the result is a subset of, or equal to, the ***domain***. The Select expression is represented by the `<Select>` element, which is of the `QuantifiedExpressionType` complex type.

```
<xs:element name="Select" type="xacml:QuantifiedExpressionType"
  substitutionGroup="xacml:Expression"/>
```

The ***iterant expression*** of a Select expression SHALL be an expression that evaluates to a value of the `http://www.w3.org/2001/XMLSchema#boolean` data-type.

The result of a Select expression SHALL be a bag of values of the same data-type as the values from the ***domain*** or “Indeterminate”.

If the ***iterant expression*** evaluates to “Indeterminate” for any value from the ***domain***, then the result of the Select expression is “Indeterminate”; otherwise, the result bag contains each value from the ***domain*** for which the ***iterant expression*** evaluates to “true”. The Select expression evaluates to an empty bag if the ***domain*** is an empty bag. Evaluation of the Select expression MAY terminate whenever the ***iterant expression*** evaluates to “Indeterminate”.

6 Functions

6.1 The attribute-designator function

The attribute-designator function produces a bag of attributes values from an ***entity*** that is either in the request context or provided as an argument. It is identified by the URI `urn:oasis:names:tc:xacml:3.0:function:attribute-designator` . When the ***entity*** is in the request context, this function emulates the `<AttributeDesignator>` element.

This function SHALL take three to five arguments. If any argument evaluates to “Indeterminate”, then the function evaluates to “Indeterminate”.

The data-type of the first argument to this function SHALL be either `http://www.w3.org/2001/XMLSchema#anyURI` or the ***entity data-type***. If the data-type is `http://www.w3.org/2001/XMLSchema#anyURI`, then the value of the argument specifies the value of the `Category` XML attribute of the ***entity*** in the request context from which the XACML attribute values will be retrieved. The value of the first argument MUST match the value of the `Category` XML attribute according to identifier equality **[XACML3]**. If the data-type is the ***entity data-type***, then the XACML attribute values SHALL be retrieved from the value of the first argument.

The second argument to this function SHALL be of data-type `http://www.w3.org/2001/XMLSchema#anyURI`. The value of the argument specifies the value of the `AttributeId` XML Attribute of the XACML attributes of the ***entity*** from which the XACML attribute values will be retrieved. The value of the second argument MUST match the value of the `AttributeId` XML attribute according to identifier equality.

The third argument SHALL be of data-type `http://www.w3.org/2001/XMLSchema#anyURI`. The value of the argument specifies the value of the `DataType` XML attribute of the XACML attribute values returned by the function. The value of the third argument MUST match the value of the `DataType` XML attribute according to identifier equality.

The fourth argument, if present, SHALL be of data-type `http://www.w3.org/2001/XMLSchema#boolean`. If the fourth argument is omitted, then it defaults to the value “false”. The value of this argument governs (in the same manner as the value of the `MustBePresent` XML attribute of the `<AttributeDesignator>` element) whether the function returns “Indeterminate” or an empty bag in the event that there are no matching attribute values to be returned. If the function returns “Indeterminate” and reports the missing attribute using a `<MissingAttributeDetail>` element and the first argument is an ***entity*** value then the value of the mandatory `Category` XML attribute of the element SHOULD be set to `urn:oasis:names:tc:xacml:3.0:attribute-category:not-applicable`. This value indicates that the attribute belongs in a ***nested entity***.

The fifth argument, if present, SHALL be of data-type `http://www.w3.org/2001/XMLSchema#string`. If the fifth argument is present, then the function MUST only return attribute values of XACML attributes with an `Issuer` XML Attribute that matches the value of the fifth argument. The value of the fifth argument MUST match the value of the `Issuer` XML attribute according to the `urn:oasis:names:tc:xacml:1.0:function:string-equal` function. If the fifth argument is absent, then all matching attribute values of the specified XACML attributes are returned without regard to the `Issuer` XML attribute.

6.1.1 Example

{Non-normative}

The two outer expressions in Figure 5 have the same effect.

```
<AttributeDesignator xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
  Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
```

```
  AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
  DataType="http://www.w3.org/2001/XMLSchema#string"
  MustBePresent="false"/>

<Apply xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
  FunctionId="urn:oasis:names:tc:xacml:3.0:function:attribute-designator">
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
    >urn:oasis:names:tc:xacml:3.0:attribute-category:resource</AttributeValue>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
    >urn:oasis:names:tc:xacml:1.0:resource:resource-id</AttributeValue>
  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.w3.org/2001/XMLSchema#string</AttributeValue>
</Apply>
```

Figure 5 - Equivalence of attribute designators

6.2 The attribute-selector function

The attribute-selector function produces a bag of unnamed and uncategorized attribute values from the `<Content>` element of an ***entity*** that is either in the request context or provided as an argument. It is identified by the URI `urn:oasis:names:tc:xacml:3.0:function:attribute-selector` . When the ***entity*** is in the request context, this function emulates the `<AttributeSelector>` element.

This function SHALL take three to five arguments. If any argument evaluates to “Indeterminate”, then the function evaluates to “Indeterminate”.

The data-type of the first argument to this function SHALL be either `http://www.w3.org/2001/XMLSchema#anyURI` or the ***entity data-type***. If the data-type is `http://www.w3.org/2001/XMLSchema#anyURI`, then the value of the argument specifies the `Category` XML attribute of the ***entity*** in the request context that contains the `<Content>` element. The value of the first argument MUST match the value of the `Category` XML attribute according to identifier equality **[XACML3]**. If the data-type is the ***entity data-type***, then the `<Content>` element of the first argument SHALL be used.

The second argument SHALL be of data-type `urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression`. The `XPathCategory` XML attribute of the value of this argument SHALL be ignored. The value contains the XPath expression to be evaluated against the `<Content>` element specified by the first argument.

The third argument SHALL be of data-type `http://www.w3.org/2001/XMLSchema#anyURI`. The value of the argument specifies the data-type of the attribute values returned from the evaluation of this function.

The fourth argument, if present, SHALL be of data-type `http://www.w3.org/2001/XMLSchema#boolean`. If the fourth argument is omitted, then it defaults to the value “false”. The value of this argument governs (in the same manner as the value of the `MustBePresent` XML attribute of the `<AttributeSelector>` element) whether the function returns “Indeterminate” or an empty bag in the event that the first argument specifies an ***entity*** in the request context that does not exist, the specified ***entity*** does not have a `<Content>` element, or the XPath expression from the second argument selects an empty node-set. If the function returns “Indeterminate” and reports the missing attribute using a `<MissingAttributeDetail>` element and the first argument is an ***entity*** value then the value of the mandatory `Category` XML attribute of the element SHOULD be set to `urn:oasis:names:tc:xacml:3.0:attribute-category:not-applicable`.

The fifth argument, if present, SHALL be of data-type `http://www.w3.org/2001/XMLSchema#anyURI`. The value of this argument specifies the `AttributeId` of an XACML attribute in the ***entity*** specified by the first argument. The referenced attribute MUST have a single value of data-type `urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression` and the XPath expression represented by that value MUST select a single node in the `<Content>` element. The `XPathCategory`

XML attribute of the value SHALL be ignored. The fifth argument corresponds to the `ContextSelectorId` XML attribute of the `<AttributeSelector>` element.

Unlike the `<AttributeSelector>` element, the attribute-selector function accepts XPath expressions that evaluate to a Boolean, string or number. If the XPath expression given by the second argument evaluates to a Boolean, then the data-type in the third argument MUST be `http://www.w3.org/2001/XMLSchema#boolean`. If the XPath expression evaluates to a string, then the data-type in the third argument MUST be `http://www.w3.org/2001/XMLSchema#string`. If the XPath expression evaluates to a number, then the data-type in the third argument MUST be `http://www.w3.org/2001/XMLSchema#double`.

The attribute-selector function is evaluated according to the following processing model, or any model that produces identical results.

1. Construct an XML data structure suitable for XPath processing from the `<Content>` element of the ***entity*** specified by the first argument. If the ***entity*** is not found or does not have a `<Content>` element, then the return value is either “Indeterminate” or an empty bag as determined by the fourth argument; otherwise, the data structure shall be constructed so that the document node of this structure contains a single document element which corresponds to the single child element of the `<Content>` element. The constructed data structure shall be equivalent to one that would result from parsing a stand-alone XML document consisting of the contents of the `<Content>` element (including any comment and processing-instruction markup).
2. Select a context node for xpath processing from this data structure. If the fifth argument is present, then the context node shall be the node selected by applying the XPath expression given in the fifth argument. It shall be an error if this evaluation returns no node or more than one node, in which case the return value MUST be “Indeterminate” with status code `urn:oasis:names:tc:xacml:1.0:status:syntax-error`. If the fifth argument is absent, then the context node shall be the document node of the data structure.
3. Evaluate the XPath expression given in the second argument using the context node selected in the previous step.
4. If the result of step 3 is a Boolean, string or number, then convert the result to a value of the data-type specified by the third argument using, respectively, the xs:boolean(), xs:string() or xs:double() constructor function from Section 5 of [XF]; otherwise (a node-set), convert the string value of each node in the result of step 3 into an XACML attribute value of the data-type specified by the third argument using the appropriate constructor function from Section 5 of **[XF]**, or as described in Section 4 in the case of the ***entity data-type***. The relevant constructor functions are:

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

   If an error occurs when converting the values returned by step 3, then the result of the function MUST be "Indeterminate" with status code `urn:oasis:names:tc:xacml:1.0:status:processing-error`.

   If the result of step 3 is an empty node-set, then the return value is either “Indeterminate” or an empty bag as determined by the fourth argument.

6.3 The entity-one-and-only function

The `urn:oasis:names:tc:xacml:3.0:function:entity-one-and-only` function SHALL take a bag of values of the ***entity data-type*** as its only argument. If the bag contains exactly one value, then the function returns that value; otherwise, the function evaluates to “Indeterminate”.

6.4 The entity-bag-size function

The `urn:oasis:names:tc:xacml:3.0:function:entity-bag-size` function SHALL take a bag of values of the ***entity data-type*** as its only argument and SHALL return an `http://www.w3.org/2001/XMLSchema#integer` value indicating the number of values in the bag.

6.5 The entity-bag function

The `urn:oasis:names:tc:xacml:3.0:function:entity-bag` function SHALL take any number of arguments of the ***entity data-type*** and return a bag containing the values of those arguments. An application of this function to zero arguments SHALL produce an empty bag of the ***entity data-type***.

7 Examples

{Non-normative}

7.1 Matching Values in a Bag

The XACML higher-order bag functions allow individual values in a bag to be operated upon, but only with pre-existing XACML functions. The ***quantified expressions*** are a generalization of the higher-order bag functions that permit arbitrarily complex expressions to be applied to the individual values of a bag.

As an example, suppose that resources have a multi-valued integer `product-code` attribute and the goal is to write an expression that is true if and only if at least one of the `product-code` values is in the range 100 to 200. A naïve solution might look like the expression in Figure 6.

```
[01] <Apply FunctionId="&xacml;1.0:function:and"
[02]   xmlns="&xacml;3.0:core:schema:wd-17">
[03]   <Apply FunctionId="&xacml;1.0:function:any-of">
[04]     <Function
[05]       FunctionId="&xacml;1.0:function:integer-greater-than-or-equal"/>
[06]     <AttributeDesignator
[07]       Category="&xacml;3.0:attribute-category:resource"
[08]       AttributeId="urn:example:xacml:product-code"
[09]       DataType="http://www.w3.org/2001/XMLSchema#integer"
[10]       MustBePresent="false"/>
[11]     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer"
[12]       >100</AttributeValue>
[13]   </Apply>
[14]   <Apply FunctionId="&xacml;1.0:function:any-of">
[15]     <Function FunctionId="&xacml;1.0:function:integer-less-than-or-equal"/>
[16]     <AttributeDesignator
[17]       Category="&xacml;3.0:attribute-category:resource"
[18]       AttributeId="urn:example:xacml:product-code"
[19]       DataType="http://www.w3.org/2001/XMLSchema#integer"
[20]       MustBePresent="false"/>
[21]     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer"
[22]       >200</AttributeValue>
[23]   </Apply>
[24] </Apply>
```

Figure 6 - Range test using higher-order bag functions

However, this doesn't work as desired. Suppose that a particular resource has the `product-code` values 50 and 250. The expression would evaluate to “true” for this resource, though it doesn't have a `product-code` value between 100 and 200, because there is no correlation between the value that satisfies the first argument of the `and` function and the value that satisfies the second argument of the `and` function. In this case, both arguments are satisfied, but by different values.

Using the ForAny expression it is possible to write an XACML expression that evaluates to “true” if and only if the `product-code` attribute has at least one value in the range 100 to 200, as in Figure 7.

```
[25] <ForAny VariableId="product-code" xmlns="&xacml;3.0:core:schema:wd-17">
[26]   <AttributeDesignator
[27]     Category="&xacml;3.0:attribute-category:resource"
[28]     AttributeId="urn:example:xacml:product-code"
[29]     DataType="http://www.w3.org/2001/XMLSchema#integer"
[30]     MustBePresent="false"/>
[31]   <Apply FunctionId="&xacml;1.0:function:and">
[32]     <Apply FunctionId="&xacml;1.0:function:integer-greater-than-or-equal">
```

```
[33]       <VariableReference VariableId="product-code"/>
[34]       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer"
[35]         >100</AttributeValue>
[36]     </Apply>
[37]     <Apply FunctionId="&xacml;1.0:function:integer-less-than-or-equal">
[38]       <VariableReference VariableId="product-code"/>
[39]       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer"
[40]         >200</AttributeValue>
[41]     </Apply>
[42]   </Apply>
[43] </ForAny>
```

Figure 7 - Range test using a quantified expression

The attribute designator beginning at line [26] extracts a bag of integer values from the `product-code` attribute. This bag becomes the ***domain*** of the ForAny expression beginning at line [25]. The ForAny expression binds each of these integer values to the `product-code` ***quantified variable*** in turn as it evaluates its ***iterant expression*** (lines [31] to [42]). The ***iterant expression*** evaluates to "true" if the value bound to the `product-code` ***quantified variable*** is both greater than or equal to 100, and less than or equal to 200. The overall expression evaluates to "true" if the ***iterant expression*** evaluates to "true" for at least one value of the ***quantified variable***.

7.2 Access Subject Relationships

The example application in this section involves the relationships an access subject has with various organizations. Each relationship is represented by a relationship ***entity***, which has the following single-valued attributes:

- `urn:example:xacml:relationship-kind`

  A string value indicating the kind of relationship the access subject has with an organization. Typical values include "`employee`", "`contractor`", "`volunteer`" and "`customer`".

- `urn:example:xacml:start-date`

  The date on which the relationship described by the ***entity*** commenced or will commence. The relationship ***entity*** is expected to be removed when the relationship terminates.

- `urn:example:xacml:organization`

  A URI value referencing the organization ***entity*** for the relationship.

The relationship ***entities*** are nested in the `urn:example:xacml:attribute:relationship` attribute of the `access-subject` category.

Each organization is represented by an organization ***entity***, which has the following attributes, among others:

- `urn:example:xacml:organization-name`

  The name of the organization.

- `urn:example:xacml:organization-type`

  A string value indicating the type of organization. Typical values include "`commercial`", "`non-profit`" and "`educational`".

The organization ***entities*** are ***related entities*** in the request context.

Figure 8 is an example of a request context making use of the relationship and organization ***entities***.

```
<Attributes xmlns="&xacml;3.0:core:schema:wd-17"
  Category="&xacml;1.0:subject-category:access-subject">
  <Attribute AttributeId="&xacml;1.0:subject:subject-id"
    IncludeInResult="false">
```



```
    <AttributeValue DataType="&xacml:1.0:data-type:rfc822Name"
      >j.smith@acme.example.com</AttributeValue>
  </Attribute>
  <Attribute AttributeId="urn:example:xacml:attribute:relationship"
             IncludeInResult="false">
    <AttributeValue DataType="&xacml;3.0:data-type:entity">
      <Attribute AttributeId="urn:example:xacml:attribute:relationship-kind"
                 IncludeInResult="false">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
          >employee</AttributeValue>
      </Attribute>
      <Attribute AttributeId="urn:example:xacml:attribute:start-date"
                 IncludeInResult="false">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#date"
          >2013-09-01</AttributeValue>
      </Attribute>
      <Attribute AttributeId="urn:example:xacml:attribute:organization"
                 IncludeInResult="false">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
          >urn:uuid:af993222-cb04-436f-a296-bbd4624e218d</AttributeValue>
      </Attribute>
    </AttributeValue>
    <AttributeValue DataType="&xacml;3.0:data-type:entity">
      <Attribute AttributeId="urn:example:xacml:attribute:relationship-kind"
                 IncludeInResult="false">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
          >customer</AttributeValue>
      </Attribute>
      <Attribute AttributeId="urn:example:xacml:attribute:start-date"
                 IncludeInResult="false">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#date"
          >2010-03-12</AttributeValue>
      </Attribute>
      <Attribute AttributeId="urn:example:xacml:attribute:organization"
                 IncludeInResult="false">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
          >urn:uuid:f182fc03-1b2b-4401-8c3d-a94675bf537e</AttributeValue>
      </Attribute>
    </AttributeValue>
  </Attribute>
</Attributes>
<Attributes xmlns="&xacml;3.0:core:schema:wd-17"
  Category="urn:uuid:af993222-cb04-436f-a296-bbd4624e218d">
  <Attribute AttributeId="urn:example:xacml:attribute:organization-name"
             IncludeInResult="false">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
      >Acme Inc.</AttributeValue>
  </Attribute>
  <Attribute AttributeId="urn:example:xacml:attribute:organization-type"
             IncludeInResult="false">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
      >commercial</AttributeValue>
  </Attribute>
</Attributes>
<Attributes xmlns="&xacml;3.0:core:schema:wd-17"
  Category="urn:uuid:f182fc03-1b2b-4401-8c3d-a94675bf537e">
  <Attribute AttributeId="urn:example:xacml:attribute:organization-name"
             IncludeInResult="false">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
      >Acmeville Building Society</AttributeValue>
  </Attribute>
  <Attribute AttributeId="urn:example:xacml:attribute:organization-type"
             IncludeInResult="false">
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
```



```
      >non-profit</AttributeValue>
  </Attribute>
</Attributes>
```

Figure 8 - Request context for access-subject and related entities

The expression in Figure 9 evaluates to “true” if and only if the access subject is a current employee of at least one non-profit organization (which is not satisfied given the preceding request context).

```
[01] <ForAny VariableId="relationship" xmlns="&xacml;3.0:core:schema:wd-17">
[02]   <AttributeDesignator
[03]     Category="&xacml;1.0:subject-category:access-subject"
[04]     AttributeId="urn:example:xacml:attribute:relationship"
[05]     DataType="&xacml;3.0:data-type:entity"
[06]     MustBePresent="false"/>
[07]   <Apply FunctionId="&xacml;1.0:function:and">
[08]     <Apply FunctionId="&xacml:1.0:function:string-equal">
[09]       <Apply FunctionId="&xacml;1.0:function:string-one-and-only">
[10]         <Apply FunctionId="&xacml;3.0:function:attribute-designator">
[11]           <VariableReference VariableId="relationship"/>
[12]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[13]             >urn:example:xacml:attribute:relationship-kind</AttributeValue>
[14]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[15]             >http://www.w3.org/2001/XMLSchema#string</AttributeValue>
[16]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean"
[17]             >true</AttributeValue>
[18]         </Apply>
[19]       </Apply>
[20]       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
[21]         >employee</AttributeValue>
[22]     </Apply>
[23]     <Apply FunctionId="&xacml;1.0:function:date-greater-than-or-equal">
[24]       <Apply FunctionId="&xacml;1.0:function:date-one-and-only">
[25]         <AttributeDesignator
[26]           Category="&xacml;3.0:attribute-category:environment"
[27]           AttributeId="&xacml;1.0:environment:current-date"
[28]           DataType="http://www.w3.org/2001/XMLSchema#date"
[29]           MustBePresent="true"/>
[30]       </Apply>
[31]       <Apply FunctionId="&xacml;1.0:function:date-one-and-only">
[32]         <Apply FunctionId="&xacml;3.0:function:attribute-designator">
[33]           <VariableReference VariableId="relationship"/>
[34]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[35]             >urn:example:xacml:attribute:start-date</AttributeValue>
[36]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[37]             >http://www.w3.org/2001/XMLSchema#date</AttributeValue>
[38]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#boolean"
[39]             >true</AttributeValue>
[40]         </Apply>
[41]       </Apply>
[42]     </Apply>
[43]     <Apply FunctionId="&xacml:3.0:function:any-of">
[44]       <Function FunctionId="&xacml:1.0:function:string-equal"/>
[45]       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
[46]         >non-profit</AttributeValue>
[47]       <Apply FunctionId="&xacml;3.0:function:attribute-designator">
[48]         <Apply FunctionId="&xacml;1.0:function:anyURI-one-and-only">
[49]           <Apply FunctionId="&xacml;3.0:function:attribute-designator">
[50]             <VariableReference VariableId="relationship"/>
[51]             <AttributeValue
[52]               DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[53]               >urn:example:xacml:attribute:organization</AttributeValue>
```



```
[54]             <AttributeValue
[55]               DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[56]               >http://www.w3.org/2001/XMLSchema#anyURI</AttributeValue>
[57]           </Apply>
[58]         </Apply>
[59]         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[60]           >urn:example:xacml:attribute:organization-type</AttributeValue>
[61]         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[62]           >http://www.w3.org/2001/XMLSchema#string</AttributeValue>
[63]       </Apply>
[64]     </Apply>
[65]   </Apply>
[66] </ForAny>
```

Figure 9 - Testing related entities of an access subject

The attribute designator beginning at line [02] generates a bag containing all the relationship ***entities*** of the access subject, which are nested in the `relationship` attribute of the `access-subject` category. This bag becomes the ***domain*** of the ForAny expression beginning at line [01]. The ForAny expression binds each of these ***entity*** values to the `relationship` ***quantified variable*** in turn as it evaluates its ***iterant expression*** (lines [07] to [65]).

The ***iterant expression*** is an `and` function with three arguments. The first argument (lines [8] to [22]) tests whether the `relationship-kind` attribute of the ***entity*** value currently bound to the `relationship` ***quantified variable*** has the string value "`employee`". The first argument to the attribute-designator function beginning at line [10] is a reference to the `relationship` ***quantified variable***, which resolves to a value of the ***entity data-type***, so the attribute-designator function obtains the `relationship` attribute from this value.

The second argument of the `and` function (lines [23] to [42]) tests whether the current date is greater than or equal to the value of the `start-date` attribute of the currently bound ***entity*** value.

The third argument of the `and` function (lines [43] to [64]) tests whether the currently bound relationship ***entity*** refers to an organization ***related entity*** with "`non-profit`" as the value of its `organization-type` attribute. The attribute-designator function beginning at line [49] obtains the URI value of the `organization` attribute of the ***entity*** value currently bound to the `relationship` ***quantified variable***. This URI becomes the value of the first argument to the attribute-designator function beginning at line [47], which attempts to find a ***related entity*** in the request context with a `Category` XML attribute matching this URI. If such an ***entity*** exists, then the `organization-type` attribute is obtained from it by the attribute-designator function.

7.3 Table-driven Policy Expression

XACML policies are sometimes used to enforce externally imposed rules or regulations. Such policies tend to display a recurring pattern where each rule or regulation is assessed against the attributes in an authorization request. As an alternative, the ***entity data-type*** can be used to represent entries in a table of rules or regulations and ***quantified expressions*** can be used to factor out the common pattern and apply it once for each entry in the table. The result is generally a policy that is more compact, easier to maintain and less error-prone because the pattern is only written once and the table entries are simple collections of XACML attributes. Changes to the rules or regulations are easily accommodated by changing attribute values in the table rather than rewriting XACML expressions. This approach is also useful in the case where policies are distributed from a central authority but need to be customized for each receiver. The customizations can be in a table managed by the receiver.

The examples in this section involve testing whether a particular type of product can be exported to a particular destination country using a table of approved exports. In Section 7.3.1 the table of approved exports is represented with XACML attributes in ***entity*** values. The examples in Section 7.3.2 show the table of approved exports represented in XML and also demonstrate the use of the attribute-selector

function. Since the table of approved exports is common to all requests it is held in an attribute in the `environment` category and would typically be fetched from a PIP by the context handler when required.

7.3.1 Table-driven Policy Expression Using XACML Attributes

The table of approved exports as a whole is held in the multi-valued `urn:example:xacml:attribute:approved-export` attribute in the `environment` category. Each entry in the table of approved exports is represented by an ***entity*** value with the following multi-valued attributes:

- `urn:example:xacml:attribute:ae-product-type`

  The string names of product-types approved for export.

- `urn:example:xacml:attribute:ae-destination`

  The approved destination countries for the associated product-types. Each destination is represented by a two-character country code.

Figure 10 shows a short example of a table of approved exports.

```
<Attribute xmlns="&xacml;3.0:core:schema:wd-17" IncludeInResult="false"
  AttributeId="urn:example:xacml:attribute:approved-export">
  <AttributeValue DataType="&xacml;3.0:data-type:entity">
    <Attribute IncludeInResult="false"
      AttributeId="urn:example:xacml:attribute:ae-product-type">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >right-handed discombobulator</AttributeValue>
    </Attribute>
    <Attribute IncludeInResult="false"
      AttributeId="urn:example:xacml:attribute:ae-destination">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >US</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >DE</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >FR</AttributeValue>
      </Attribute>
  </AttributeValue>
  <AttributeValue DataType="&xacml;3.0:data-type:entity">
    <Attribute IncludeInResult="false"
      AttributeId="urn:example:xacml:attribute:ae-product-type">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >left-handed discombobulator</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >left-handed combobulator</AttributeValue>
    </Attribute>
    <Attribute IncludeInResult="false"
      AttributeId="urn:example:xacml:attribute:ae-destination">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >AU</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
        >GB</AttributeValue>
    </Attribute>
  </AttributeValue>
</Attribute>
```

Figure 10 - The table of approved exports as an attribute

The expression in Figure 11 evaluates to “true” if and only if the product type specified by the `product-type` attribute in the `resource` category is approved for export to the destination specified by the `destination` attribute in the `action` category according to the table of approved exports.

```
[01] <ForAny VariableId="approved-export" xmlns="&xacml;3.0:core:schema:wd-17">
[02]   <AttributeDesignator
[03]     Category="&xacml;3.0:attribute-category:environment"
[04]     AttributeId="urn:example:xacml:attribute:approved-export"
[05]     DataType="&xacml;3.0:data-type:entity"
[06]     MustBePresent="false"/>
[07]   <Apply FunctionId="&xacml;1.0:function:and">
[08]     <Apply FunctionId="&xacml;1.0:function:string-is-in">
[09]       <Apply FunctionId="&xacml;1.0:function:string-one-and-only">
[10]         <AttributeDesignator
[11]           Category="&xacml;3.0:attribute-category:resource"
[12]           AttributeId="urn:example:xacml:attribute:product-type"
[13]           DataType="http://www.w3.org/2001/XMLSchema#string"
[14]           MustBePresent="false"/>
[15]       </Apply>
[16]       <Apply FunctionId="&xacml;3.0:function:attribute-designator">
[17]         <VariableReference VariableId="approved-export"/>
[18]         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[19]           >urn:example:xacml:attribute:ae-product-type</AttributeValue>
[20]         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[21]           >http://www.w3.org/2001/XMLSchema#string</AttributeValue>
[22]       </Apply>
[23]     </Apply>
[24]     <Apply FunctionId="&xacml;1.0:function:string-is-in">
[25]       <Apply FunctionId="&xacml;1.0:function:string-one-and-only">
[26]         <AttributeDesignator
[27]           Category="&xacml;3.0:attribute-category:action"
[28]           AttributeId="urn:example:xacml:attribute:destination"
[29]           DataType="http://www.w3.org/2001/XMLSchema#string"
[30]           MustBePresent="false"/>
[31]       </Apply>
[32]       <Apply FunctionId="&xacml;3.0:function:attribute-designator">
[33]         <VariableReference VariableId="approved-export"/>
[34]         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[35]           >urn:example:xacml:attribute:ae-destination</AttributeValue>
[36]         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[37]           >http://www.w3.org/2001/XMLSchema#string</AttributeValue>
[38]       </Apply>
[39]     </Apply>
[40]   </Apply>
[41] </ForAny>
```

Figure 11 - Testing if export is approved

The attribute designator beginning at line [02] generates an ***entity*** bag containing the two attribute values of the `approved-export` attribute from the `environment` category. This bag becomes the ***domain*** of the ForAny expression beginning at line [01]. The ForAny expression binds each of these ***entity*** values to the `approved-export` ***quantified variable*** in turn as it evaluates the ***iterant expression*** (lines [07] to [40]). The attribute-designator function beginning at line [16] extracts the values of the `ae-product-type` attribute from the ***entity*** value currently bound to the `approved-export` ***quantified variable***, returning them as a bag of "string" values to be compared with the value of the `product-type` attribute from the `resource` category by the `string-is-in` function on line [08]. The attribute-designator function beginning at line [32] extracts the values of the `ae-destination` attribute from the ***entity*** value currently bound to the `approved-export` ***quantified variable***, returning them as a bag of "string" values to be compared with the value of the `destination` attribute from the `action` category by the `string-is-in` function on line [24].

Since there are two values in the domain, the ***iterant expression*** will be evaluated at most twice (noting that the ForAny expression can stop as soon as it finds a value of the ***quantified variable*** for which the

iterant expression evaluates to “true”). For one of these evaluations, the attribute-designator beginning at line [16] will return a bag containing the string value “`right-handed discombobulator`” and the attribute-designator function beginning at line [32] will return a bag containing the string values “`US`”, “`DE`” and “`FR`”. For the other evaluation of the ***iterant expression***, the attribute-designator beginning at line [16] will return a bag containing the string values “`left-handed discombobulator`” and “`left-handed combobulator`” and the attribute-designator function beginning at line [32] will return a bag containing the string values “`AU`” and “`GB`”.

Representing the table of approved exports as an environment attribute allows for it to be maintained in, or generated from, an external data source and fetched on demand by a PIP. As an environment attribute it is also available for use by any rule, policy or policy set. If the table of approved exports is not externally maintained and is only used within one XACML policy, then the table could alternatively be represented as a variable definition of that policy, with the ***domain*** of the ForAny expression replaced with a reference to that variable. If the table is only used within the ***domain*** of one ***quantified expression***, then the ***domain*** could be that table directly.

The expression in Figure 11 expects an authorization request to contain exactly one value for the `product-type` attribute and exactly one value for the `destination` attribute (see the `string-one-and-only` functions on lines [09] and [25]). The expression in Figure 12 is a reformulation of the expression in Figure 11 that allows a request to contain multiple values for the `product-type` and/or multiple values for the `destination`. This expression evaluates to “true” if and only if every `product-type` in the request is approved for export to every `destination` in the request.

```
[42] <ForAll VariableId="product-type" xmlns="&xacml;3.0:core:schema:wd-17">
[43]   <AttributeDesignator
[44]     Category="&xacml;3.0:attribute-category:resource"
[45]     AttributeId="urn:example:xacml:attribute:product-type"
[46]     DataType="http://www.w3.org/2001/XMLSchema#string"
[47]     MustBePresent="false"/>
[48]   <ForAll VariableId="destination">
[49]     <AttributeDesignator
[50]       Category="&xacml;3.0:attribute-category:action"
[51]       AttributeId="urn:example:xacml:attribute:destination"
[52]       DataType="http://www.w3.org/2001/XMLSchema#string"
[53]       MustBePresent="false"/>
[54]     <ForAny VariableId="approved-export">
[55]       <AttributeDesignator
[56]         Category="&xacml;3.0:attribute-category:environment"
[57]         AttributeId="urn:example:xacml:attribute:approved-export"
[58]         DataType="&xacml;3.0:data-type:entity"
[59]         MustBePresent="false"/>
[60]       <Apply FunctionId="&xacml;1.0:function:and">
[61]         <Apply FunctionId="&xacml;1.0:function:string-is-in">
[62]           <VariableReference VariableId="product-type"/>
[63]           <Apply FunctionId="&xacml;3.0:function:attribute-designator">
[64]             <VariableReference VariableId="approved-export"/>
[65]             <AttributeValue
[66]               DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[67]               >urn:example:xacml:attribute:ae-product-type</AttributeValue>
[68]             <AttributeValue
[69]               DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[70]               >http://www.w3.org/2001/XMLSchema#string</AttributeValue>
[71]           </Apply>
[72]         </Apply>
[73]         <Apply FunctionId="&xacml;1.0:function:string-is-in">
[74]           <VariableReference VariableId="destination"/>
[75]           <Apply FunctionId="&xacml;3.0:function:attribute-designator">
[76]             <VariableReference VariableId="approved-export"/>
[77]             <AttributeValue
[78]               DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[79]               >urn:example:xacml:attribute:ae-destination</AttributeValue>
```



```
[80]             <AttributeValue
[81]               DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[82]               >http://www.w3.org/2001/XMLSchema#string</AttributeValue>
[83]           </Apply>
[84]         </Apply>
[85]       </Apply>
[86]     </ForAny>
[87]   </ForAll>
[88] </ForAll>
```

Figure 12 - Testing if export is approved for multiple products and destinations

The ForAll expression beginning at line [42] has the `product-type` attribute from the `resource` category as its ***domain***. Thus it binds each value of that attribute to its `product-type` ***quantified variable*** as it evaluates its ***iterant expression*** (lines [48] to [87]). The ForAll expression beginning at line [48] has the `destination` attribute from the `action` category as its ***domain***. Thus it binds each value of that attribute to its `destination` ***quantified variable*** as it evaluates its ***iterant expression*** (lines [54] to [86]). Between them the two ForAll expressions consider every combination of `product-type` and `destination` in the request. The ForAny expression beginning at line [54] determines whether the current values bound to the `product-type` and `destination` ***quantified variables*** match at least one of the ***entity*** values from the `approved-export` environment attribute. The overall expression evaluates to “true” if the ForAny expression evaluates to “true” for every combination of `product-type` and `destination`.

7.3.2 Table-driven Policy Expression Using XML

The example in this section addresses the same application as the previous section except that the table of approved exports is instead represented as XML in the `<Content>` element of a single ***entity*** value of the `urn:example:xacml:attribute:approved-exports` attribute in the `environment` category as shown in Figure 13.

```
[01] <Attribute xmlns="&xacml;3.0:core:schema:wd-17" IncludeInResult="false"
[02]   AttributeId="urn:example:xacml:attribute:approved-exports">
[03]   <AttributeValue DataType="&xacml;3.0:data-type:entity">
[04]     <Content>
[05]       <ae:ApprovedExports xmlns:ae="urn:example:approved-export">
[06]         <ae:ApprovedExport>
[07]           <ae:ProductType>right-handed discombobulator</ae:ProductType>
[08]           <ae:Destination>US</ae:Destination>
[09]           <ae:Destination>DE</ae:Destination>
[10]           <ae:Destination>FR</ae:Destination>
[11]         </ae:ApprovedExport>
[12]         <ae:ApprovedExport>
[13]           <ae:ProductType>left-handed discombobulator</ae:ProductType>
[14]           <ae:ProductType>left-handed combobulator</ae:ProductType>
[15]           <ae:Destination>AU</ae:Destination>
[16]           <ae:Destination>GB</ae:Destination>
[17]         </ae:ApprovedExport>
[18]       </ae:ApprovedExports>
[19]     </Content>
[20]   </AttributeValue>
[21] </Attribute>
```

Figure 13 - The table of approved exports as XML

In this case, the expression to test whether a particular product type can be exported to a particular destination country makes use of the attribute-selector function instead of the attribute-designator function, as shown in Figure 14.

```
[22] <ForAny VariableId="approved-export"
[23]   xmlns="&xacml;3.0:core:schema:wd-17"
[24]   xmlns:ae="urn:example:approved-export">
[25]   <Apply FunctionId="&xacml;3.0:function:attribute-selector">
[26]     <Apply FunctionId="&xacml;3.0:function:entity-one-and-only">
[27]       <AttributeDesignator
[28]         Category="&xacml;3.0:attribute-category:environment"
[29]         AttributeId="urn:example:xacml:attribute:approved-exports"
[30]         DataType="&xacml;3.0:data-type:entity"
[31]         MustBePresent="false"/>
[32]     </Apply>
[33]     <AttributeValue
[34]       DataType="&xacml;3.0:data-type:xpathExpression"
[35]       XPathCategory="urn:example:xacml:category:ignored"
[36]       >ae:ApprovedExports/ae:ApprovedExport</AttributeValue>
[37]     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[38]       >&xacml;3.0:data-type:entity</AttributeValue>
[39]   </Apply>
[40]   <Apply FunctionId="&xacml;1.0:function:and">
[41]     <Apply FunctionId="&xacml;1.0:function:string-is-in">
[42]       <Apply FunctionId="&xacml;1.0:function:string-one-and-only">
[43]         <AttributeDesignator
[44]           Category="&xacml;3.0:attribute-category:resource"
[45]           AttributeId="urn:example:xacml:attribute:product-type"
[46]           DataType="http://www.w3.org/2001/XMLSchema#string"
[47]           MustBePresent="false"/>
[48]       </Apply>
[49]       <Apply FunctionId="&xacml;3.0:function:attribute-selector">
[50]         <VariableReference VariableId="approved-export"/>
[51]         <AttributeValue
[52]           DataType="&xacml;3.0:data-type:xpathExpression"
[53]           XPathCategory="urn:example:xacml:category:ignored"
[54]           >ae:ApprovedExport/ae:ProductType</AttributeValue>
[55]         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[56]           >http://www.w3.org/2001/XMLSchema#string</AttributeValue>
[57]       </Apply>
[58]     </Apply>
[59]     <Apply FunctionId="&xacml;1.0:function:string-is-in">
[60]       <Apply FunctionId="&xacml;1.0:function:string-one-and-only">
[61]         <AttributeDesignator
[62]           Category="&xacml;3.0:attribute-category:action"
[63]           AttributeId="urn:example:xacml:attribute:destination"
[64]           DataType="http://www.w3.org/2001/XMLSchema#string"
[65]           MustBePresent="false"/>
[66]       </Apply>
[67]       <Apply FunctionId="&xacml;3.0:function:attribute-selector">
[68]         <VariableReference VariableId="approved-export"/>
[69]         <AttributeValue
[70]           DataType="&xacml;3.0:data-type:xpathExpression"
[71]           XPathCategory="urn:example:xacml:category:ignored"
[72]           >ae:ApprovedExport/ae:Destination</AttributeValue>
[73]         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
[74]           >http://www.w3.org/2001/XMLSchema#string</AttributeValue>
[75]       </Apply>
[76]     </Apply>
[77]   </Apply>
[78] </ForAny>
```

Figure 14 - Testing if export is approved using the attribute-selector function

The attribute designator beginning at line [27] extracts the ***nested entity*** (line [03] to line [20]) from the `approved-exports` attribute, which becomes the first argument to the attribute-selector function beginning at line [25]. This attribute-selector function generates an ***entity*** bag containing the two `<ApprovedExport>` child elements as separate ***entity*** values, as illustrated in Figure 15.

```
<Apply FunctionId="&xacml;3.0:function:entity-bag">
  <AttributeValue DataType="&xacml;3.0:data-type:entity">
    <Content>
      <ae:ApprovedExport xmlns:ae="urn:example:approved-export">
        <ae:ProductType>right-handed discombobulator</ae:ProductType>
        <ae:Destination>US</ae:Destination>
        <ae:Destination>DE</ae:Destination>
        <ae:Destination>FR</ae:Destination>
      </ae:ApprovedExport>
    </Content>
  </AttributeValue>
  <AttributeValue DataType="&xacml;3.0:data-type:entity">
    <Content>
      <ae:ApprovedExport xmlns:ae="urn:example:approved-export">
        <ae:ProductType>left-handed discombobulator</ae:ProductType>
        <ae:ProductType>left-handed combobulator</ae:ProductType>
        <ae:Destination>AU</ae:Destination>
        <ae:Destination>GB</ae:Destination>
      </ae:ApprovedExport>
    </Content>
  </AttributeValue>
</Apply>
```

Figure 15 - The table of approved exports as an entity bag

The ForAny expression beginning at line [22] binds each of these ***entity*** values to the `approved-export` ***quantified variable*** in turn as it evaluates the ***iterant expression*** (lines [40] to [77]). The attribute-selector function beginning at line [49] extracts the values of the `<ProductType>` child elements from the ***entity*** currently bound to the `approved-export` ***quantified variable***, returning them as a bag of “string” values to be compared with the value of the `product-type` attribute from the request context. The attribute-selector function beginning at line [67] extracts the values of the `<Destination>` child elements from the ***entity*** currently bound to the `approved-export` ***quantified variable***, returning them as a bag of “string” values to be compared with the value of the `destination` attribute from the request context.

8 Security Considerations

Entities may contain sensitive information that must be protected from unauthorized disclosure. A policy writer interacting with a policy administration point (PAP) would not normally see any ***entities*** known to the PIP or PEP, but there are ways a policy writer could manipulate policies in order to discover the values of ***entity*** attributes.

The most direct method is to use attribute assignment expressions in obligations and advice **[XACML3]**. Attribute assignment expressions provide a means to return information from the request context to the PEP. This attack requires the existence of an obligation or advice that presents the extracted attribute values in a form that is accessible to the policy writer. For example, an obligation that can be co-opted to send an email to the policy writer, display an on-screen message in an application that the policy writer uses or print a message in a log file that the policy writer can read. The policy writer adds such an obligation or advice to a policy that is under the writer's control. The policy writer can then cause the targeted information to be extracted by instigating an access attempt for which the policy is applicable, or by just waiting until such an access attempt occurs in the course of normal operations.

A more indirect method of exposing ***entity*** attributes is to change a target or condition in a policy under the control of the policy writer so that an authorization decision is dependent on the value of an ***entity*** attribute the policy writer is attempting to discover. Using relational functions (`type-less-than`, `type-equal` and `type-greater-than`) and a binary search, the policy writer can home in on the actual value of the ***entity*** attribute by iteratively editing the condition and observing the effect on the authorization decision.

These methods are available using only the core capabilities of XACML, however, the attribute-designator function defined in this profile allows access to ***entities*** that might not otherwise be accessible through the request context.

If a policy writer is a highly privileged user with access to the ***entity*** data stores underlying the PIP and PEP, then the fact that the policy writer can obtain information about ***entities*** by other means is not an issue. However, there are two ways a less-privileged user can inject a malicious policy.

The `<XACMLAuthzDecisionQuery>` element **[SAML]** allows an XACML client to include additional policies to be used by the PDP in evaluating the authorization request. A malicious client (or a client that has been compromised by a malicious user) could provide a policy that is designed to discover ***entity*** attributes.

The Administration and Delegation Profile **[ADMIN]** defines a mechanism for the delegation of policy administration. A delegate can create arbitrary policies, but the applicability of those policies is limited in scope by administrative policies created by trusted policy writers. However, the procedure for establishing the authority of a delegate's policy does not take into consideration the obligations and advice in the policy. There is no verification of the obligations and advice and therefore no restriction on the attribute assignment expressions the delegate uses. The procedure also does not restrict which functions or attributes the delegate can use in a policy. The delegate could create a policy that is applicable within the delegated scope, but is designed to discover ***entity*** attributes.

To protect ***entity*** attributes from unauthorized disclosure, implementers might consider the following strategies:

- Disallow policies and policy sets in `<XACMLAuthzDecisionQuery>` elements.
- Only accept policies and policy sets in `<XACMLAuthzDecisionQuery>` elements provided by authenticated, trusted clients.
- Disallow the attribute-designator function in policies and policy sets provided in `<XACMLAuthzDecisionQuery>` elements. Note that this still leaves the contents of the predefined ***entities*** such as the access subject and resource open to discovery.
- Apply access controls to the ***entity*** attributes accessed by policies and policy sets in `<XACMLAuthzDecisionQuery>` elements, or otherwise limit the ***entities*** and ***entity*** attributes that these policies and policy sets can access.

With respect to delegation, implementers might consider the following mitigation strategies:

- Disable or not implement the capabilities of the Administration and Delegation Profile.
- Disallow the attribute-designator function in policies and policy sets written by delegates. Note that this still leaves the contents of the predefined ***entities*** open to discovery.
- Have the PAP limit the ***entity*** attributes that delegates can reference in their policies.
- Apply access controls to the ***entity*** attributes accessed by delegate's policies and policy sets during reduction **[ADMIN]** by the PDP.

A malicious policy writer could use ***quantified expressions*** to deliberately create policies that perform excessive computation resulting in a reduction of service or denial of service for applications using the PDP. ***Quantified expressions*** should be considered in whatever methods are used to mitigate denial of service attacks.

9 Conformance

An implementation claiming conformance with this specification MUST support the ***entity data-type*** defined in Section 4, the ***quantified expressions*** defined in Section 5 and all the functions defined in Section 6 except for the attribute-selector function. Support for the attribute-selector function is OPTIONAL.

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:
Steven Legg, ViewDS Identity Solutions
Hal Lockhart, Individual
Erik Rissanen, Axiomatics
Mohammad Jafari, Veterans Health Administration
Rich Levinson, Oracle
John Tolbert, Queralt, Inc.

Voting members of the XACML Technical Committee:

Person	Organization	Role
Bill Parducci	Individual	Chair
Hal Lockhart	Individual	Chair
Steven Legg	ViewDS Identity Solutions	Member

Appendix B. Revision History

Revision	Date	Editor	Changes Made
WD 01	22 Oct 2013	Steven Legg	Initial draft.
WD 02	11 June 2015	Steven Legg	All the changes from the previous draft are editorial in nature. The content of the introduction has been rearranged into three sections: a new, briefer, non-normative overview with two high-level diagrams, a non-normative section on the background and a normative section on the specifics of related and nested entities. The section on the entity data type has been moved ahead of the section on quantified expressions. The XML Schema has been moved to a separate artifact.
WD 03	8 July 2015	Steven Legg	The return value for the attribute-selector function when the entity is absent or the `<Content>` element is absent has been specified.
WD 04	13 August 2020	Steven Legg	The base type for the `QuantifiedExpressionType` XML Schema type has been corrected in this profile and in the associated XML Schema file. The references have been updated to newer versions. The XML entity reference used in examples has been properly named in Section 1.3. Missing default namespace declarations in Figure 8 have been added. Spurious double spaces have been removed.
WD 05	20 August 2020	Steven Legg	The JSON equivalent representation in figures 3 and 4 has changed from a JSON member to a JSON value (i.e., removed the “Attribute”: prefix). The JSON profile now requires an array value for “Attribute”, but array notation here would be confusing.
WD 06	13 September 2023	Steven Legg	Defined a `not-applicable` attribute category URI for use in the `Category` XML attribute of a `<MissingAttributeDetail>` element. The JSON profile now requires an array value for the “Value” member. Figures 3 and 4 have been corrected.

			A number of example `<AttributeDescriptor>` elements using start-tag notation have been changed to the correct empty-element tag notation. Fixed the `xPathExpression` values in Figure 14. Some literal values contained non-breaking hyphens instead of ASCII minus. *All* hyphens in values have been changed to make sure they are all minus characters.