Attribute Manifest File, v0.8
July 16, 2009

Contributors
Hal Lockhart, Oracle Corporation
Anil Tappetla, Cisco

0.1	Hal Lockhart	Initial Draft
0.2	Hal Lockhart	Added <Field> element Added various sections
0.5	Hal Lockhart	Add Contributor list Add <Scopes> element Split <Key> into <KeyXACMLName> and <KeyRepositoryName> Allow multiple instances of Key Add examples in section 4
0.8	Hal Lockhart	Formatting for OASIS Contribution

1 Introduction

This document defines a format, called an Attribute manifest File (AMF), for communicating metadata about information, in the form of attributes which may be used in making access control policy decisions. It specifies an XML format for exchanging this data. It describes several usecases in which this data might be consumed. It also suggests how the data might be generated. However, it does not specify an specific processing algorithms or system architectures for its generation or consumption.

Access control policy languages, such as [XACML] feature a policy language syntax and processing rules for evaluation as well as a format for providing necessary input data and referencing it from the policy language. The language syntax and processing rules are encapsulated from all other components of the access control system and protected applications. The actual policies evaluated by a given deployed system are also encapsulated from all other components except for the tools used to construct them. And in fact, these systems may not even have the ability to evaluate policies, merely to produce them from human input or alternative policy representations.

This encapsulation has a number of benefits, discussed elsewhere, but it leads to a conundrum. In order for policies to be evaluated and an access control decision made, the input data must be provided. Rich, powerful languages are capable of making use of any sort of data available at the time to control the decision. However this data must be provided at the time of the decision by software components which are unaware of the policies currently in force or the rules for evaluating them. Of course the PDP can make attribute requests to obtain the values referenced by policies which are missing from the request, however this will increase decision latency and usually reduce overall throughput of the PDP.

If there was no cost providing information or if the total range of inputs to policy decisions was small, callers could simply provide all available information with each request. Many authorization decision systems operate in this way today. This specification defines a data format which should allow callers of a PDP to provide information more closely tailored to policy needs, while not breaking the encapsulation of the PDP.

1.1 Namespaces

Prefix	Namespace	Specification(s)
amf	{TBD}	This specification

1.2 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [KEYWORDS].

When describing concrete XML schemas, this specification uses the notational convention of WS-Security. Specifically, each member of an element's [children] or [attributes] property is described using an XPath-like notation (e.g., /x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element wildcard (<xs:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xs:anyAttribute/>).

1.3 Normative References

[WS-MEX] W3C First Public Working Draft, Web Services Metadata Exchange (WS-MetadataExchange), 17 March 2009, http://www.w3.org/TR/2009/WD-ws-metadata-exchange-20090317/

[WS-POLICY] W3C Recommendation "Web Services Policy 1.5 – Framework", 04 September 2007 http://www.w3.org/TR/2007/REC-ws-policy-20070904/

[WS-SP] OASIS Standard, **"**WS-SecurityPolicy 1.2", July 2007 http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702

[XACML] T. Moses, ed., OASIS eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS Standard, 1 February 2005, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

2 Usecases

There are several ways in which attribute information could be used by access control system components other than a PDP.

2.1 Configuring a PIP

A Policy Information Point (PIP) is a system component which is responsible for obtaining attribute information and making it available to a Context Handler for inclusion in a Request Context. In a many environments, basic attribute information, such as Subject Identifier or Resource Name. However, additional information such as the Subject's Groups or Roles or other Resource properties my have to be obtained from other sources.

A PIP needs to know what attributes it should provide. It also needs to know where this information is located. It also needs to know how to retrieve the attributes, typically by using previously provided attribute values as lookup keys. In a typical scenario, PIPs located at various points in the system can be configured with AMFs which allow them to examine an existing Request Context and add additional attributes from other sources. This is illustrated in figure 1.

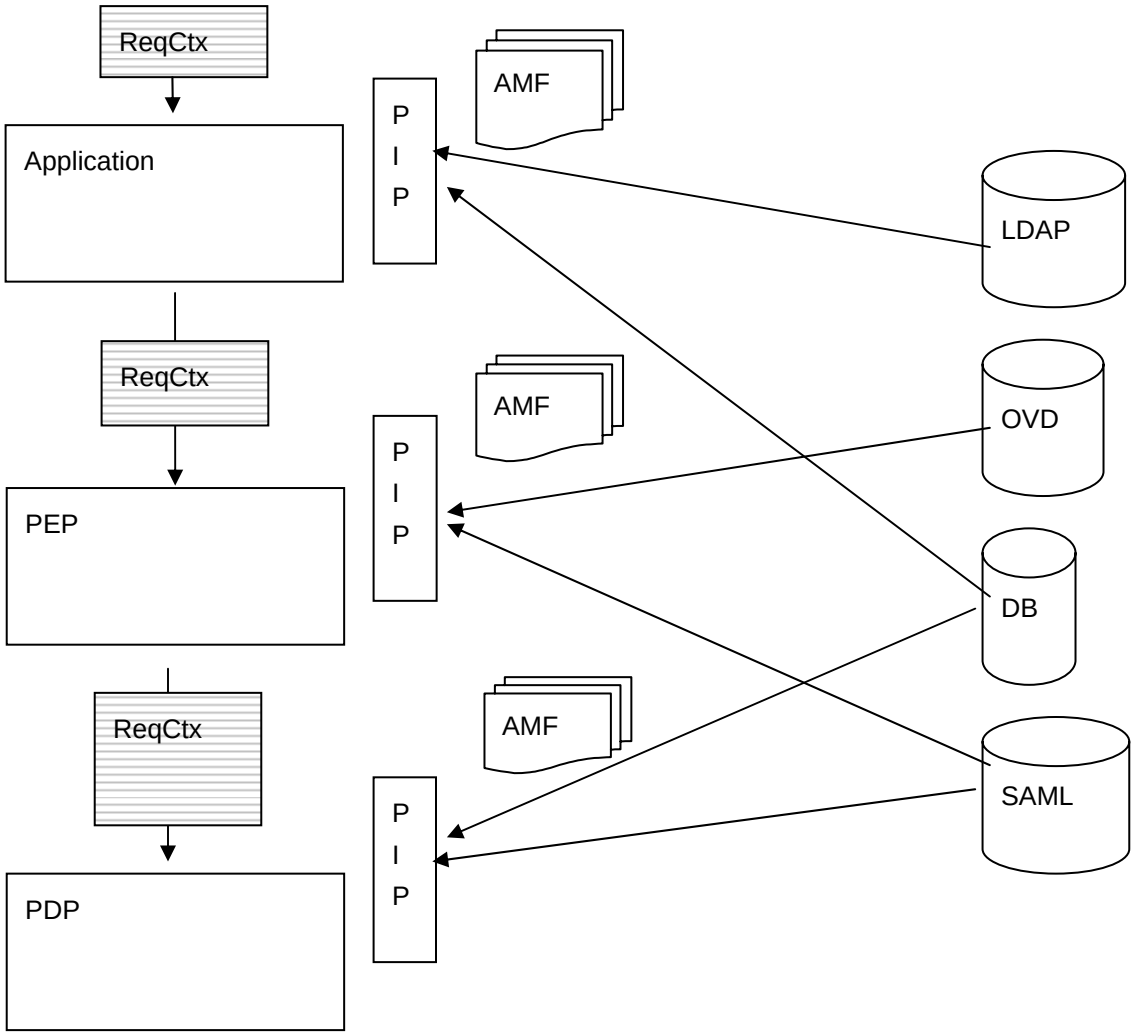

Figure 1 – Attribute Enhancement

2.2 Handling Missing attributes

One or more PIPs may also be configured to obtain attribute information only after the PDP has determined it is needed for the current request. This can be done in two general ways. The [XACML] standard specifies that if a PDP is unable to evaluate the policies relating to a request because some necessary attribute information has not been supplied, it can return a result of Indeterminate and specify what attributes are missing. Since the PDP will usually not know how to obtain these attributes, the PIP can use the AMF to retrieve them and provide them to the Context Handler so the decision request can be repeated with the missing attributes added. Note that just like the previous use case, the PIP might be configured at various points in the architecture. This is illustrated in figure 2.

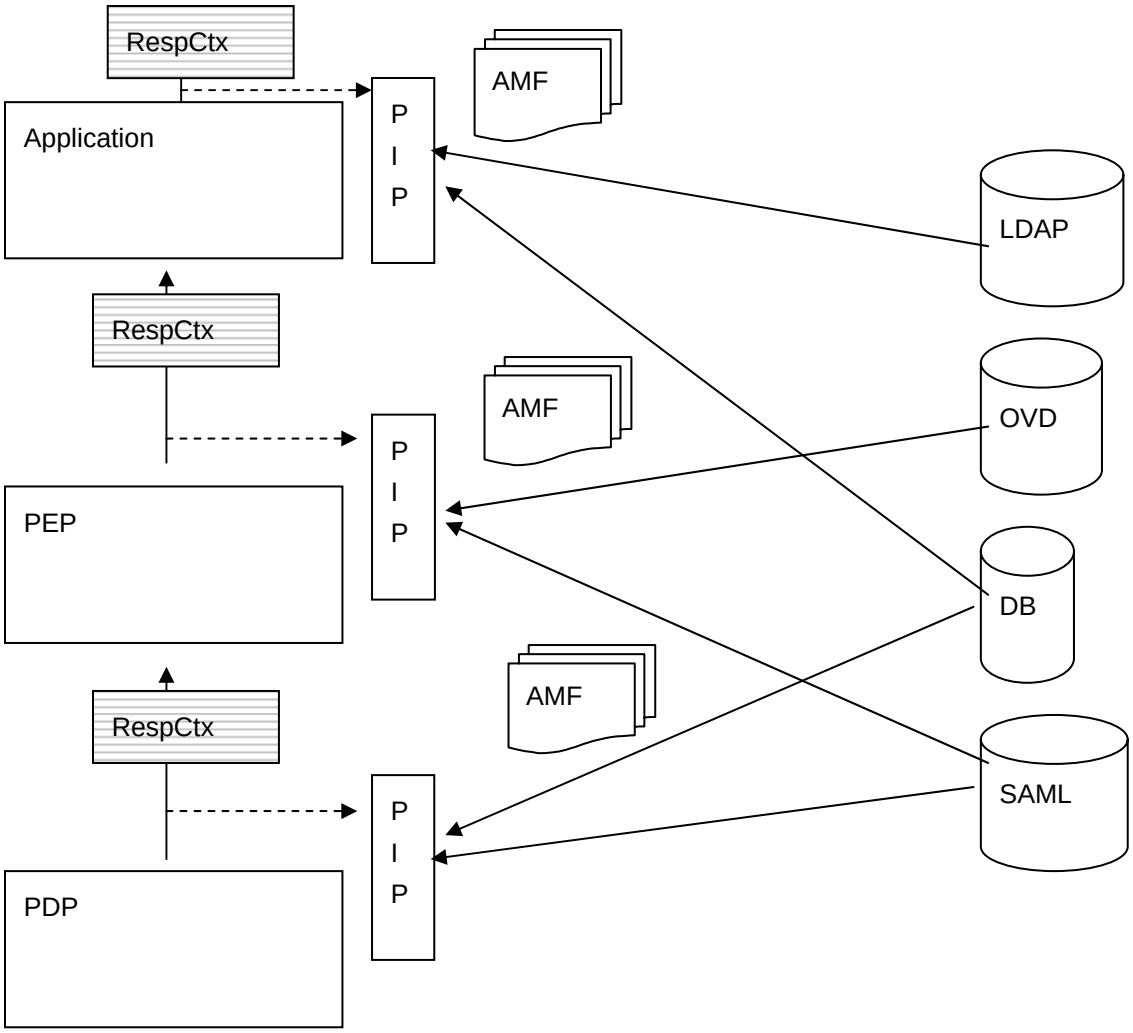

Figure 2 – Handling Missing Attributes

As an alternative a PDP may be implemented with an attributed request handler which can obtain missing attributes while policy evaluation is occurring without the need to return an Indeterminate with missing attributes result. This possibility is discussed in a non-normative section of [XACML], but no specific formats, protocols or interfaces are defined to enable it. Nevertheless, a number of XACML PDP implementations allow for it. Its advantages in reducing decision latency are obvious. An internal attribute request handler could also make use of an AMF to determine how to obtain the attribute information in question.

2.3 Vocabulary Advertisement

Another usecase for an AMF is as a mechanism for a PDP (or application service) to advertise what attributes are used in policy decisions. In this case, the retrieval information associated with the attributes might or might not be used. The AMF could be attached to a [WS-XACML] policy, security policy [WS-SP] or other [WS-POLICY] based language. Alternative it could be exchanged using [WS-MEX] or other mechanism.

2.4 Policy Management Tools

Tools for editing or analyzing XACML policies might also make use of AMFs. In most cases, only the naming and type information would be used, not the retrieval information. For example, a policy editor might use the list of attribute names and/or legal values or value ranges to populate drop down menus. A policy analysis tool might use the information to step through a series of policy decision variants in order report which requests would be allowed under various attributes and their values.

2.5 AMF Creation

An AMF could be generated in a variety of ways.

- Manually using a text editor or XML tool
- Mechanically by scanning a set of policies for the attributes they reference
- Resource attributes might be obtained by scanning application program declarations
- Subject attributes might be obtained by scanning repositories such as directories and databases of user information

The AMF is intended as an exchange format. Most likely individual attribute information will be split apart or combined as PIP and other points of consumption are configured.

3 Format

The AMF consists of the following syntax.

```
<amf:NeededAttributes>
<amf:NeededAttribute amf:Autoload=”true” >
  <amf:AttributeId> … </amf:AttributeId>
  <amf:Issuer> … </amf:Issuer>
  <amf:Category> … </amf:Category>
  <amf:Datatype> … </amf:Datatype>
  <amf:Scopes> … </amf:Scopes>
  <amf:EnumeratedValues> … </amf:EnumeratedValues>
  <amf:Range>
    <amf:Minimum> … </amf:Minimum>
    <amf:Maximum> … </amf:Maximum>
  <amf:/Range>
  <amf:RetrievalInfo>
    <amf:Location> … </amf:Location>
    <amf:Method> … </amf:Method>
    <amf:RetrievalURI> … </amf:RetrievalURI>
    <amf:Key>
      <amf:KeyXACMLName> … </amf:KeyXACMLName>
      <amf:KeyRepositoryName> … <amf:KeyRepositoryName>
    </amf:Key>
    <amf:Field … <amf:Field>
  <amf:/RetrevialInfo>
<amf:/NeededAttribute>
<amf:NeededAttribute>
   …
<amf:/NeededAttribute>
  …
<amf:/NeededAttributes>
```

The following defines the elements and attributes listed in the schema overview above.

/amf:NeededAttributes

This mandatory element contains one of more <NeededAttribute> element.

/amf:NeededAttributes/amf:NeededAttribute

This element contains the information about a single attribute that may be used as input to an authorization policy decision. This element may appear any number of times.

/amf:NeededAttributes/amf:NeededAttribute/@Autoload

This optional Boolean attribute indicates whether a PIP should retrieve the attribute and put it in the Request Context (assuming the necessary key is available) automatically or only upon receiving a Missing Attributes error or other indication.

/amf:NeededAttributes/amf:NeededAttribute/amf:AttributeId

This mandatory element contains a URI indicating the XACML AttributeId of the XACML Attribute. (Note that AttributeId is an XML Attribute in the XACML Schema.)

/amf:NeededAttributes/amf:NeededAttribute/amf:Issuer

This optional element contains a string indicating the XACML Issuer of the XACML Attribute. (Note that Issuer is an XML Attribute in the XACML Schema.)

/amf:NeededAttributes/amf:NeededAttribute/amf:Category

This optional element contains a URI indicating the XACML Category of the XACML Attribute. Legal values are the eight values defined by XACML, plus **FIXME** SameAsRetrievalKey, which indicates the Category should be set to the value of the Category of the XACML Attribute used to retrieve this Attribute's value. (Note that Category is an XML Attribute in the XACML Schema.)

/amf:NeededAttributes/amf:NeededAttribute/amf:Datatype

This mandatory element contains a URI indicating the XACML datatype. Its value MUST be one of the XACML defined dataypes. (Note that Datatype is an XML Attribute in the XACML Schema.)

/amf:NeededAttributes/amf:NeededAttribute/amf:Scopes

This optional element contains a list of identifiers of Scopes that this Attribute is a member.

/amf:NeededAttributes/amf:NeededAttribute/amf:EnumeratedValues

This optional element contains a list of legal values for this XACML attribute separated by a comma or new line character.

/amf:NeededAttributes/amf:NeededAttribute/amf:Range

This optional element indicates the minimum and/or maximum legal values of the XACML attribute. The datatype of the attribute must be one for which minimum and maximum values can be defined.

/amf:NeededAttributes/amf:NeededAttribute/amf:Range/amf:Minimum

This optional element indicates the minimum legal value of the XACML attribute. The type must be the same as the datatype of the XACML attribute.

/amf:NeededAttributes/amf:NeededAttribute/amf:Range/amf:Maximum

This optional element indicates the maximum legal value of the XACML attribute. The type must be the same as the datatype of the XACML attribute.

/amf:NeededAttributes/amf:NeededAttribute/amf:RetrievalInfo

This optional element contains information which may be used to retrieve the XACML attribute value from where it is stored.

/amf:NeededAttributes/amf:NeededAttribute/amf:RetrievalInfo/amf:Location

This optional element of type string indicates the network location of the repository where the XACML attribute value is stored.

/amf:NeededAttributes/amf:NeededAttribute/amf:RetrievalInfo/amf:Method

This optional element of type string indicates the protocol or access method to be used to obtain the XACML attribute value.

/amf:NeededAttributes/amf:NeededAttribute/amf:RetrievalInfo/amf:RetrievalURI

This optional element contains a URI which may be dereferenced to obtain the XACML attribute value. Depending on the type of URI and access method, it may be necessary to insert the key value (below) into the URI at an appropriate point.

/amf:NeededAttributes/amf:NeededAttribute/amf:RetrievalInfo/amf:Key

This optional element contains elements identifying an XACML attribute contained in the current Request Context (or otherwise available to the PIP) the value of which may be used as a key to retrieve the value of the needed attribute. For example, username might be used as a key to obtain the group attribute. This element may occur any number of times. Each key will be used in the order given.

/amf:NeededAttributes/amf:NeededAttribute/amf:RetrievalInfo/amf:Key/amf:KeyXACMLName

This element contains the AttributeId of the XACML attribute to be used as a key.

/amf:NeededAttributes/amf:NeededAttribute/amf:RetrievalInfo/amf:Key/amf:KeyRepositoryName

This element contains the name of the attribute as used by the Repository or Authority if different from the XACML AttributeId.

/amf:NeededAttributes/amf:NeededAttribute/amf:RetrievalInfo/amf:Field

This optional element contains the name of the field in which the attribute value is stored in the repository if it is different from the XACML AtttributeId.

4 Examples

This section profiles the use of AMF in different contexts.

4.1 Type Information Only

This example shows an AMF with only type information. This would be suitable for use by a Policy Editing tool.

```
<amf:NeededAttributes>
<amf:NeededAttribute>
  <amf:AttributeId> Username </amf:AttributeId>
  <amf:Category>
    urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject
  </amf:Category>
  <amf:Datatype>
    http://www.w3.org/2001/XMLSchema#string
  </amf:Datatype>
<amf:/NeededAttribute>

<amf:NeededAttribute>
  <amf:AttributeId> Department </amf:AttributeId>
  <amf:Category> urn:oasis:names:tc:xacml:1.0:subject-
category:access-subject
  </amf:Category>
  <amf:Datatype> http://www.w3.org/2001/XMLSchema#string
  </amf:Datatype>
  <amf:EnumeratedValues>
    Manufacturing
    Research and Development
    Marketing and Sales
    Finance and Administration
  </amf:EnumeratedValues>
<amf:/NeededAttribute>

<amf:/NeededAttributes>
```

4.2 LDAP

This example shows how AMF would be used to retrieve an attribute from an LDAP directory.

```
<amf:NeededAttributes>
<amf:NeededAttribute>
  <amf:AttributeId> Department </amf:AttributeId>
  <amf:Category> urn:oasis:names:tc:xacml:1.0:subject-
category:access-subject
```

```
  </amf:Category>
  <amf:Datatype> http://www.w3.org/2001/XMLSchema#string
  </amf:Datatype>
  <amf:RetrievalInfo>
    <amf:Location> ldap.example.com </amf:Location>
    <amf:Method> LDAP </amf:Method>
    <amf:Key>
      <amf:KeyXACMLName>
        urn:oasis:names:tc:xacml:1.0:subject:subject-id
      </amf:KeyXACMLName>
      <amf:KeyRepositoryName> CN <amf:KeyRepositoryName>
    </amf:Key>
  <amf:/RetrevialInfo>
<amf:/NeededAttribute>

<amf:/NeededAttributes>
```

The PIP retrieves the value of the user's Department from the directory at ldap.example.com. The value of the subject id in the request context is used as the CN. The attribute received in this case is called Department in LDAP and in the Request Context.

4.3 SQL

This example shows how AMF would be used to retrieve an attribute using SQL.

```
<amf:NeededAttributes>
<amf:NeededAttribute amf:Autoload="true" >
  <amf:AttributeId> Doctype </amf:AttributeId>
  <amf:Category>
    urn:oasis:names:tc:xacml:3.0:attribute-category:resource
  </amf:Category>
  <amf:Datatype>
    http://www.w3.org/2001/XMLSchema#string
  </amf:Datatype>
  <amf:RetrievalInfo>
    <amf:Location>
      dbserver.example.com/documentinfo
    </amf:Location>
    <amf:Method> SQL </amf:Method>
    <amf:Key>
      <amf:KeyXACMLName>
        urn:oasis:names:tc:xacml:1.0:resource:resource-id
      </amf:KeyXACMLName>
      <amf:KeyRepositoryName> Title <amf:KeyRepositoryName>
    </amf:Key>
  <amf:/RetrevialInfo>
<amf:/NeededAttribute>

<amf:/NeededAttributes>
```

In this example the PIP will retrieve the value of the Doctype attribute from the database and add it to the Request Context, because of the Autoload=”true” attribute. The value s found in the database located at dbserver.example.com in the documentinfo table. The PIP loads the XACML Resource ID from the Request Context into a variable called Resource. It then uses this SQL statement to retrieve the correct record.

```
SELECT FROM documentinfo WHERE Title = Resource
```

It then puts the value of the Doctype field in the Request Context as a resource attribute of type string.

4.4 SAML Attribute Authority

This example shows how AMF would be used to retrieve an attribute from a SAML Attribute Authority using the SAML .

```
<amf:NeededAttributes>
<amf:NeededAttribute>
  <amf:AttributeId> Group </amf:AttributeId>
  <amf:Category> SameAsRetrievalKey </amf:Category>
  <amf:Datatype>
    http://www.w3.org/2001/XMLSchema#string
  </amf:Datatype>
  <amf:RetrievalInfo>
    <amf:Location> aa.example.com </amf:Location>
    <amf:Method> SAML </amf:Method>
    <amf:Key>
      <amf:KeyXACMLName>
        urn:oasis:names:tc:xacml:1.0:subject:subject-id
      </amf:KeyXACMLName>
      <amf:KeyRepositoryName>
        Subject/NameID
      <amf:KeyRepositoryName>
    </amf:Key>
  <amf:/RetrevialInfo>
<amf:/NeededAttribute>

<amf:/NeededAttributes>
```

The PIP examines the Request Context for attributes of type subject id. For each it does a SAML Attribute Query to retrieve the Group attribute. The server is aa.example.com. The Subject Name Id in the query is the value of the subject id from that subject category.

Each Group attribute is put in the Request Context with a type of string and a category of the same type as the subject id used for that query.

Appendix A. XACML Values

- ## XACML Category types

  - ### Core Types

```
These are defined by XACML.

urn:oasis:names:tc:xacml:3.0:attribute-category:resource
urn:oasis:names:tc:xacml:3.0:attribute-category:action
urn:oasis:names:tc:xacml:3.0:attribute-category:environment
urn:oasis:names:tc:xacml:1.0:subject-category:access-subject
urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject
urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject
urn:oasis:names:tc:xacml:1.0:subject-category:codebase
urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine
```

  - ### AMF Types

This is used to indicate that the Category should be copied from the Key Attribute.

{TBD} SameAsRetrievalType

- ## XACML DataTypes

```
urn:oasis:names:tc:xacml:1.0:data-type:x500Name.
urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name
urn:oasis:names:tc:xacml:2.0:data-type:ipAddress
urn:oasis:names:tc:xacml:2.0:data-type:dnsName
urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression
http://www.w3.org/2001/XMLSchema#string
http://www.w3.org/2001/XMLSchema#boolean
http://www.w3.org/2001/XMLSchema#integer
http://www.w3.org/2001/XMLSchema#double
```

```
http://www.w3.org/2001/XMLSchema#time
http://www.w3.org/2001/XMLSchema#date
http://www.w3.org/2001/XMLSchema#dateTime
http://www.w3.org/2001/XMLSchema#anyURI
http://www.w3.org/2001/XMLSchema#hexBinary
http://www.w3.org/2001/XMLSchema#base64Binary
urn:oasis:names:tc:xacml:2.0:data-type:dayTimeDuration
urn:oasis:names:tc:xacml:2.0:data-type:yearMonthDuration
```