XACML-BASED PRIVACY POLICY LANGUAGES

Anne Anderson
Senior Staff Engineer
Sun Microsystems, Inc.

Some General Requirements

- Avoid conflicting standards
  - XACML and XACML Privacy Profile are approved OASIS Standards in widespread use[1]
- Combine access control and privacy policies at the enforcement level
  - Often deal with the same resources; keep consistent
  - XACML does this
- Support Obligations
  - XACML does this
- Support negotiation
  - XACML derivatives do this

XACML Constraint Language

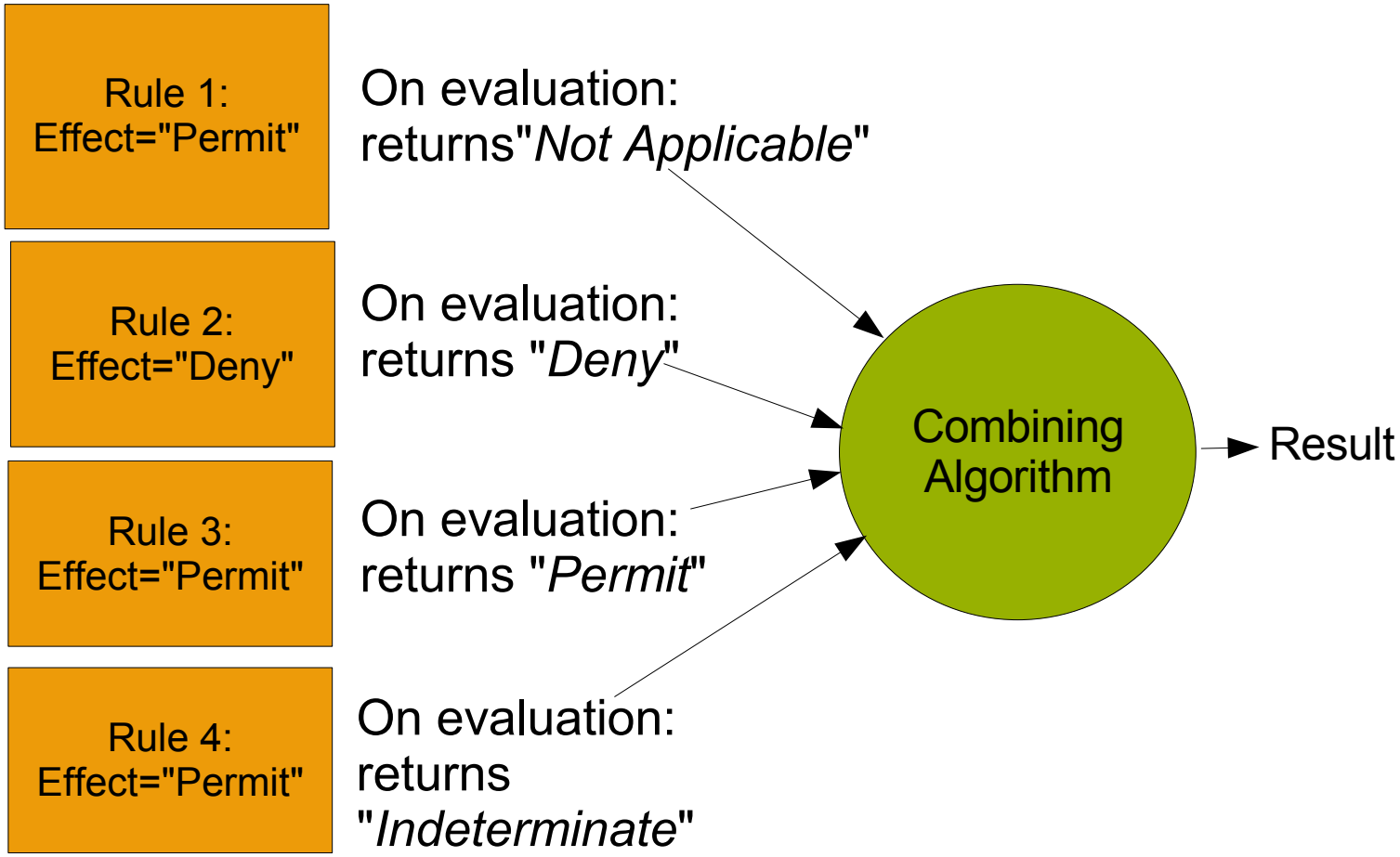
Policy fragment: *"The P3P retention options STP (stated purpose), LEG (legal requirement), and BUS (business practice) are acceptable"*

```
<!-- Returns true iff the first arg is a subset of the 2nd arg -->
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-subset">

   <!-- 1st arg: returns values of all Attributes with "retention" AttributeId -->
   <AttributeDesignator DataType=".../XMLSchema#string"
                        AttributeId="urn:...:p3p:retention"/>

   <!-- 2nd argument: converts a collection of values into a "bag" -->
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
      <!-- The compact retention values  that are acceptable -->
      <AttributeValue DataType=".../XMLSchema#string">STP<AttributeValue>
      <AttributeValue DataType=".../XMLSchema#string">LEG<AttributeValue>
      <AttributeValue DataType=".../XMLSchema#string">BUS<AttributeValue>
   <Apply>
<Apply>
```

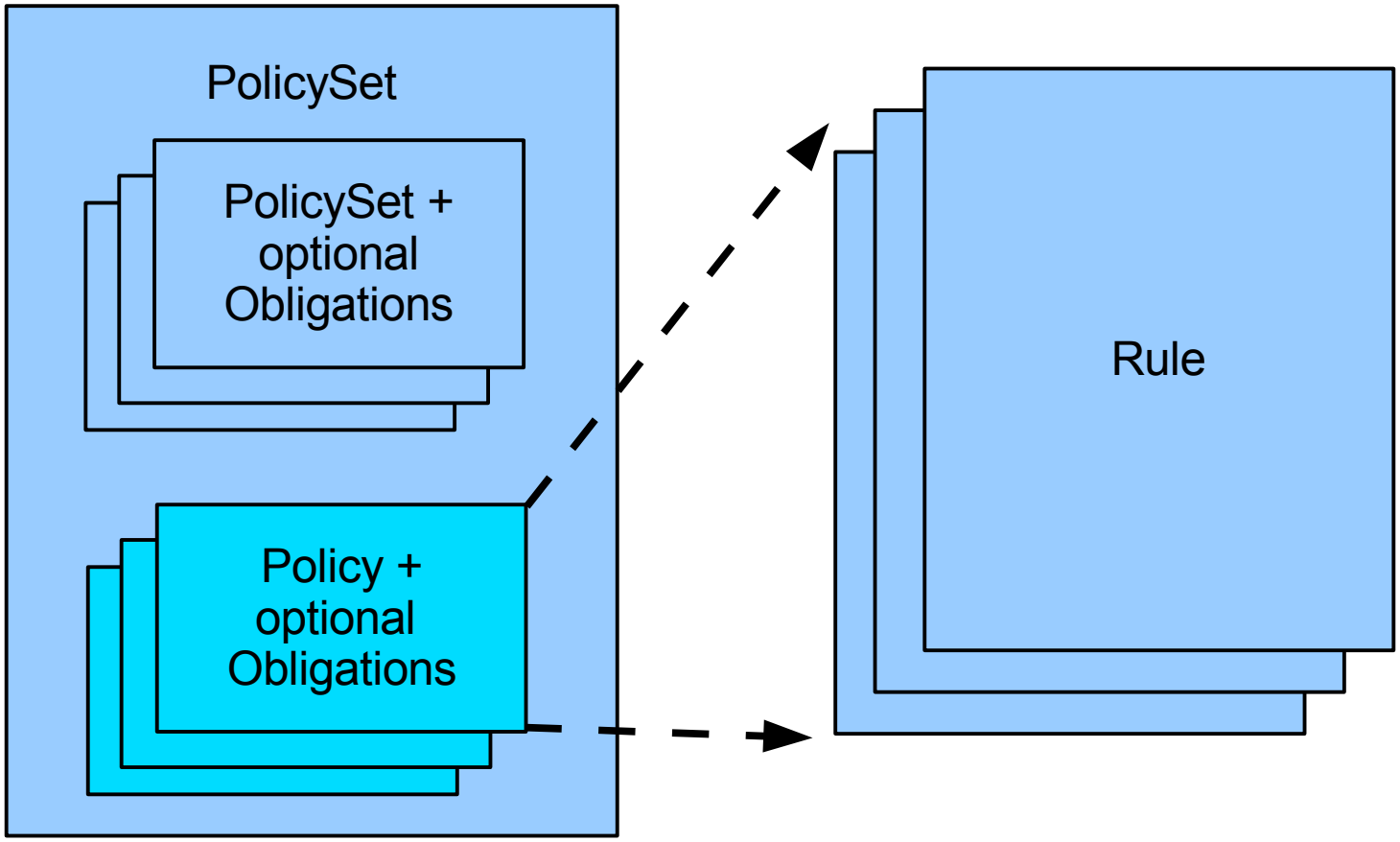
XACML Rules

```
<Rule RuleId="RetentionRule"  Effect="Permit">

   <Target>
      ...optional simple pre-condition constraints to efficiently determine applicability
      ...e.g.  Applies if requester is a member of "Consortium XYZ"
   </Target>

   <Condition>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">

         ...constraint functions...

         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">

            ...constraint functions...

         </Apply>
      </Apply>
   </Condition>

</Rule>
```

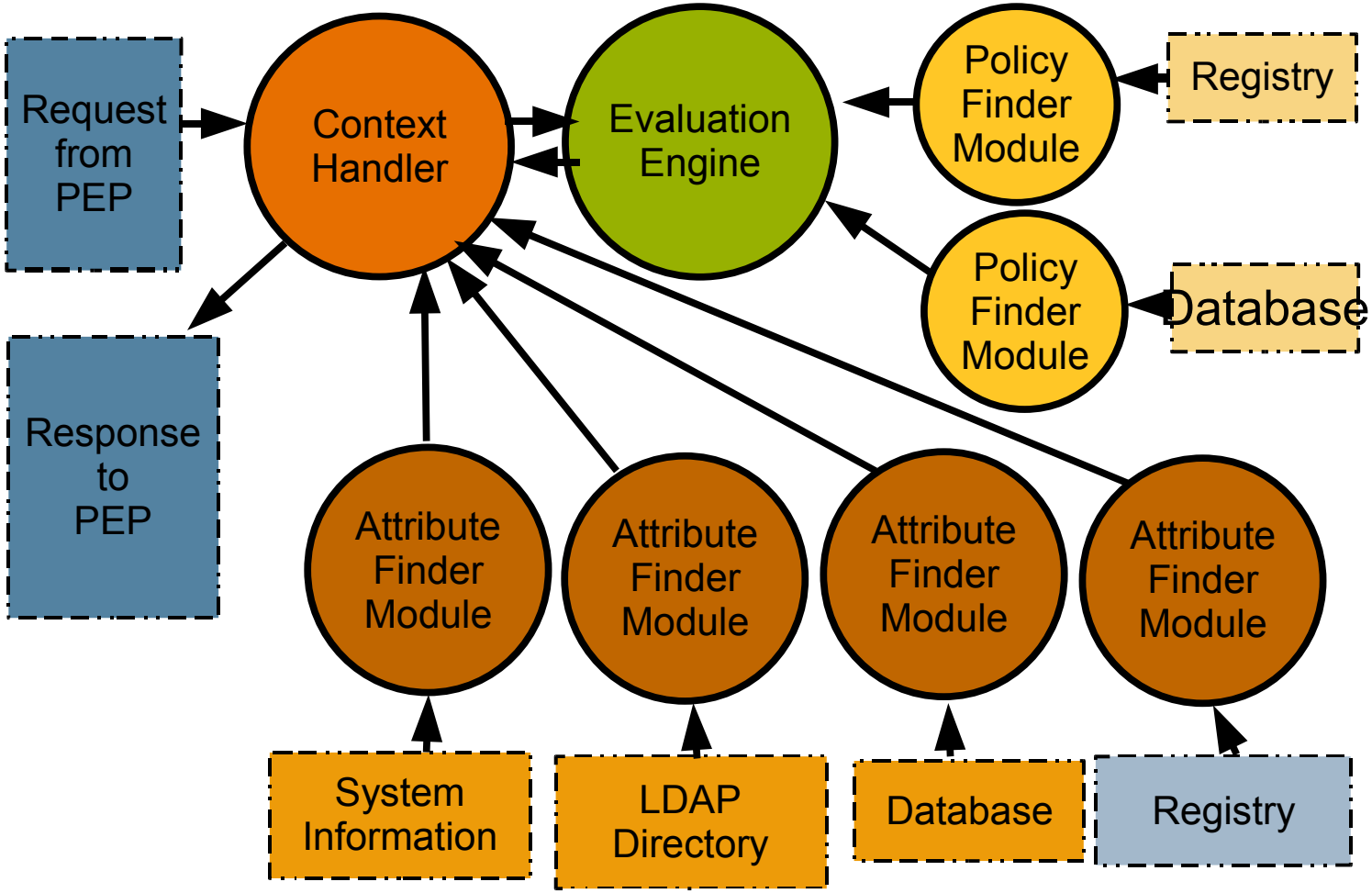
XACML Combining Algorithms



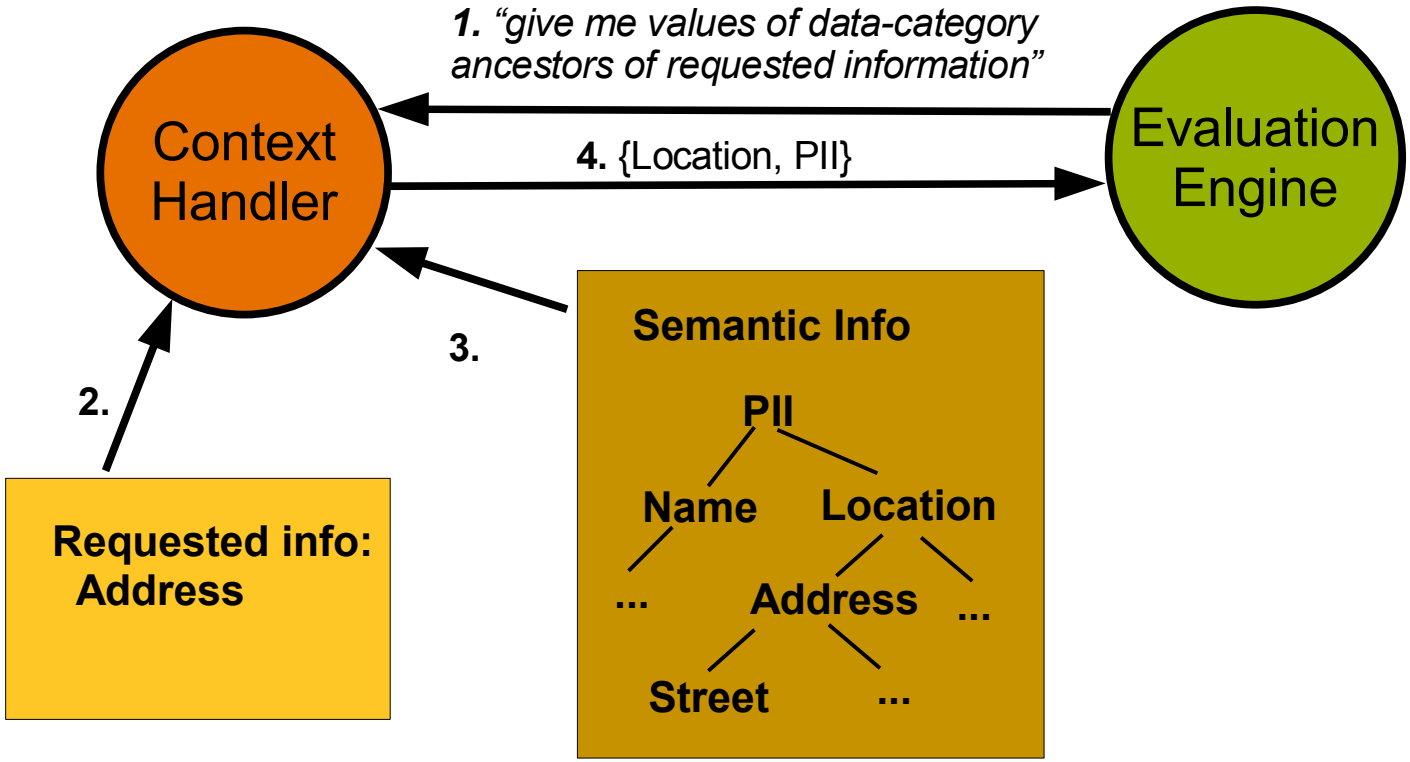
XACML Policies and PolicySets



Applying Semantic Information to XACML Policies



Applying Semantic Information to XACML Policies



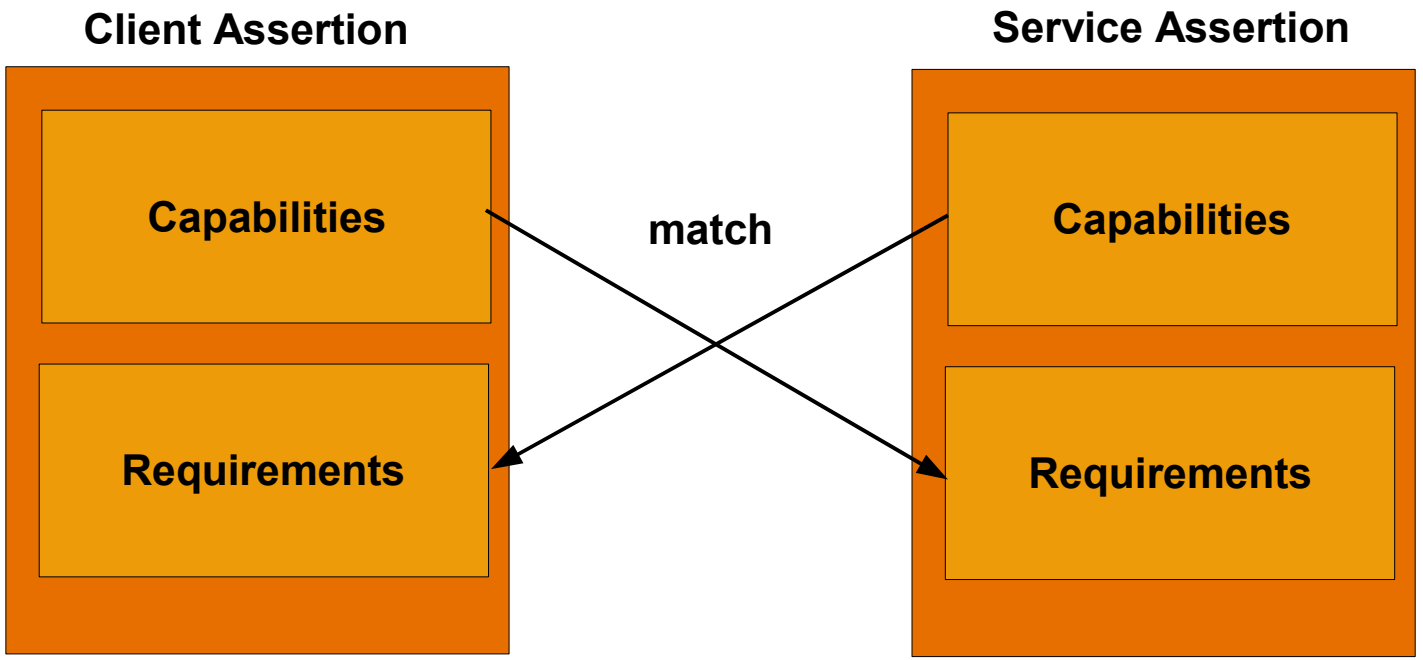
Matching Client and Service Policies

- "XACML profile for Web-services (WSPL)"
  - Subset of XACML and functions with complete intersections defined
- "XACML-Based Web Services Policy Constraint Language (WS-PolicyConstraints)"
  - Just the functions and intersections from WSPL
- "Domain-Independent Policy Assertion Language (DIPAL)"
  - Attempt to standardize WS-PolicyConstraints
- "Web Services Profile of XACML (WS-XACML)"
  - Intersectable authorization, access control, and privacy policy assertions for use in WS-Policy

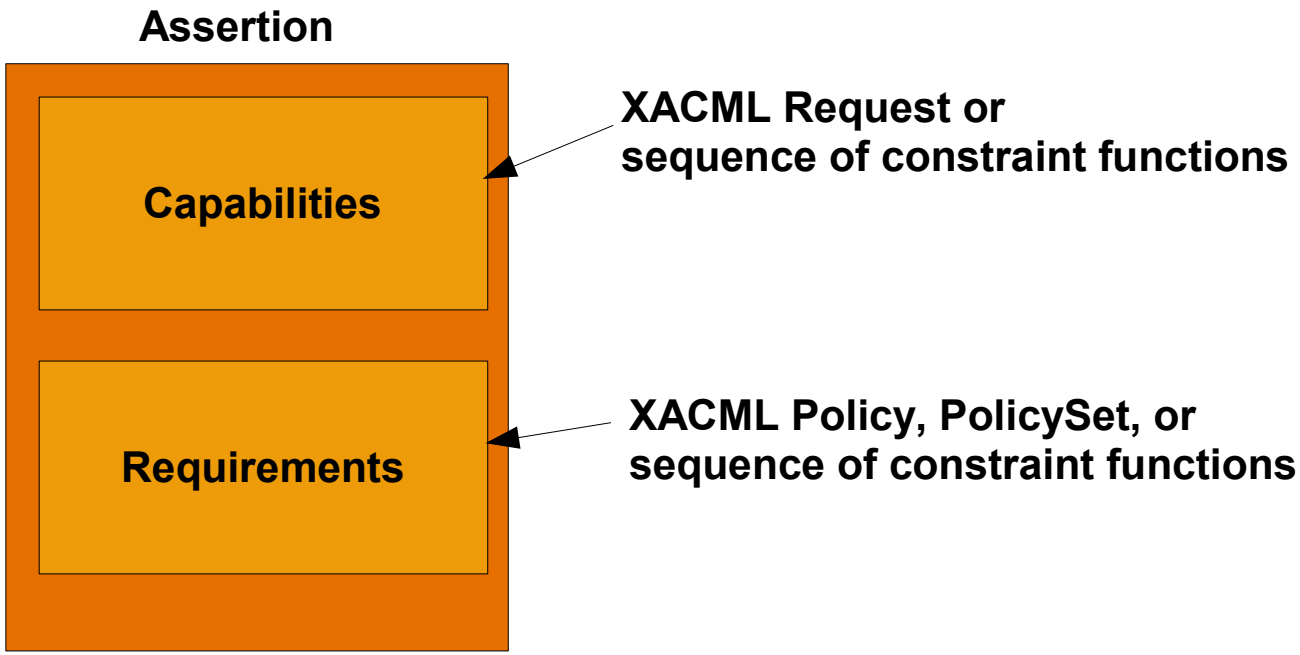
Matching Examples

- Client: x ≥ 30 Service: x = 10
  - Intersection empty – incompatible requirements/capabilities
- Client: x ≥ 30 Service: x = 35
  - Intersection: 35
  - This satisfies both policies
- Client: x in {A, B, C} Service: x in {B, C}
  - Intersection: x in {B, C}
  - Either value satisfies both policies
- All based on semantics of datatypes and arithmetic/sets
  - Usually a floor or ceiling operation or set intersection

WS-XACML



WS-XACML



References

[1] XACML References: Products and Deployments
http://docs.oasis-open.org/xacml/xacmlRefs.html#Products

[2] XACML Profile for Web Services (WS-XACML)
draft 5, 9 October 2006
http://www.oasis-open.org/committees/download.php/20643/xacml-3.0-profile-webservices-spec-v1.0-wd-5-en.pdf

Sun, Sun Microsystems, the Sun logo, and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and in other countries.

XACML-BASED PRIVACY POLICY LANGUAGES

Anne Anderson
anne.anderson@sun.com

XACML-BASED PRIVACY POLICY LANGUAGES

Anne Anderson
Senior Staff Engineer
Sun Microsystems, Inc.

Some General Requirements

- Avoid conflicting standards
  - > XACML and XACML Privacy Profile are approved OASIS Standards in widespread use[1]
- Combine access control and privacy policies at the enforcement level
  - > Often deal with the same resources; keep consistent
  - > XACML does this
- Support Obligations
  - > XACML does this
- Support negotiation
  - > XACML derivatives do this

XACML Constraint Language

Policy fragment: *"The P3P retention options STP (stated purpose), LEG (legal requirement), and BUS (business practice) are acceptable"*

```
<!-- Returns true iff the first arg is a subset of the 2nd arg -->
<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-subset">

  <!-- 1st arg: returns values of all Attributes with "retention" AttributeId -->
  <AttributeDesignator DataType=".../XMLSchema#string"
                       AttributeId="urn:...:p3p:retention"/>

  <!-- 2nd argument: converts a collection of values into a "bag" -->
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
    <!-- The compact retention values  that are acceptable -->
    <AttributeValue DataType=".../XMLSchema#string">STP<AttributeValue>
    <AttributeValue DataType=".../XMLSchema#string">LEG<AttributeValue>
    <AttributeValue DataType=".../XMLSchema#string">BUS<AttributeValue>
  <Apply>
<Apply>
```


Page 3

The heart of the XACML language is a type-based constraint language. Policies are expressed as constraint functions over variables that represent the policy vocabulary. These variables are called Attributes. Functions in a policy rule are evaluated against the values for the variables that describe a specific access request. The variables are just identifiers, and are extensible; XACML defines a few standard ones such as "subject-id", "resource-id", "action-id", and "current-time". XPath expressions may also be used as variable identifiers, allowing policies to express constraints on the contents of XML documents, for example fields in a hospital patient record that a user wants to access. Each variable has a specified datatype. There are 14 standard XACML datatypes, including integer, string, boolean, double, date, dateTime, URI, hexBinary, base64Binary, RFC 822 and X.500 names. Datatypes are also just identifiers, and are extensible, but extended functions must be created to support any extended datatypes – in practice, we are not aware of anyone who has found a need to create new datatypes, however. There are about 28 standard XACML functions: for each of these, there are versions for each of the standard datatypes that make sense for those functions. The functions include various equality and comparison functions, regular-expression matching functions, arithmetic functions (including ones for dates and times), string manipulation functions, "bag" functions for dealing with bags of variable values, set functions, higher-order bag functions such as "any-of-all" and "all-of-any", and a few XPath-based functions such as returning a count of the nodes selected by a given XPath expression.

XACML's semantics are based on the well-understood semantics for these generic datatypes and functions.

XACML's Privacy Profile, which is part of the XACML 2.0 OASIS Standard, defines standard variables to represent purposes: the purpose for which data was gathered, and the purpose for which data is being accessed, and illustrates how to create a constraint that requires these to be consistent.

XACML Rules

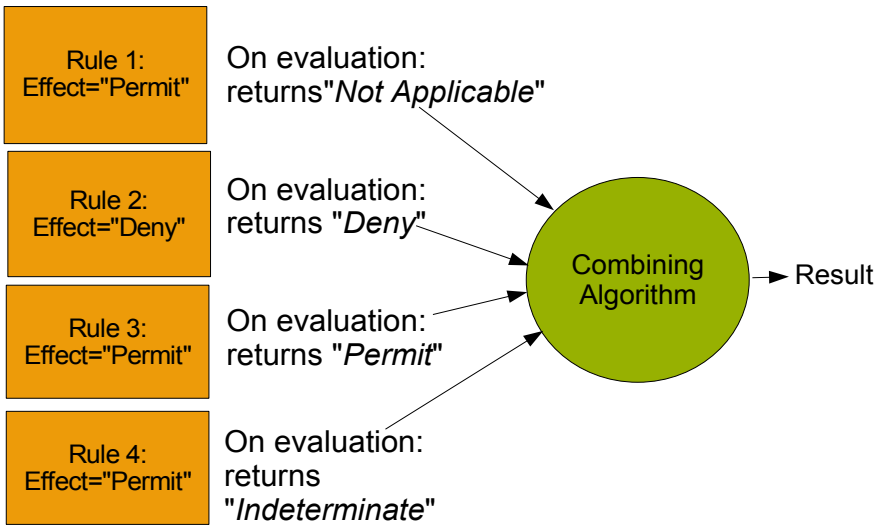
```
<Rule RuleId="RetentionRule"  Effect="Permit">

  <Target>
    ...optional simple pre-condition constraints to efficiently determine applicability
    ...e.g.  Applies if requester is a member of "Consortium XYZ"
  </Target>

  <Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">

      ...constraint functions...

      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">

        ...constraint functions...

      </Apply>
    </Apply>
  </Condition>

</Rule>
```


A collection of constraints is expressed in an XACML Rule. Simple constraints that help determine the requests to which this Rule applies may be separated out into a "Target" - this allows for more efficient evaluation and for indexing of rules. More complex constraints go into the "Condition". In the Condition, constraints may be combined using Boolean operators, and data manipulation functions such as arithmetic may be applied to variables.

A Rule is satisfied if both its Target and its Condition evaluate to "True" when evaluated against the Attributes associated with a specific access request.

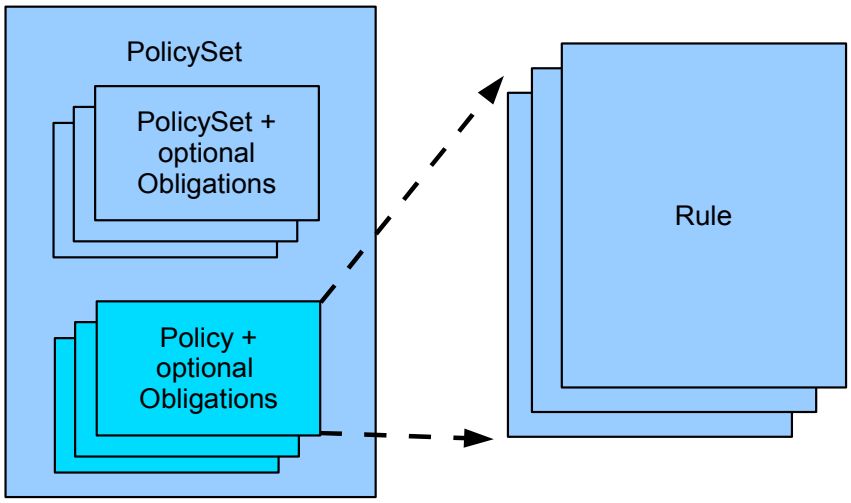
Each Rule has an "Effect" of Permit or Deny. This is the value to be returned from evaluating the Rule if the Rule's Target and Condition are satisfied. If either the Target or the Condition is "false", then the Rule returns a value of "Not Applicable". If an error occurs in the evaluation of the Target or the Condition, a value of "Indeterminate" is returned.

XACML Combining Algorithms



Since an XACML Policy may contain many Rules, and since various Rules might return different values for the same access request, there must be a way of resolving conflicts in their results. This is done by specifying a “Combining Algorithm” for each Policy. There are several standard Combining Algorithms, but these are extensible. A typical Combining Algorithm is “Deny Overrides”: if any Rule returns “Deny”, then the Policy returns “Deny”. Only if all Rules return either “Permit” or “Not Applicable” does the Policy return “Permit”.

XACML Policies and PolicySets



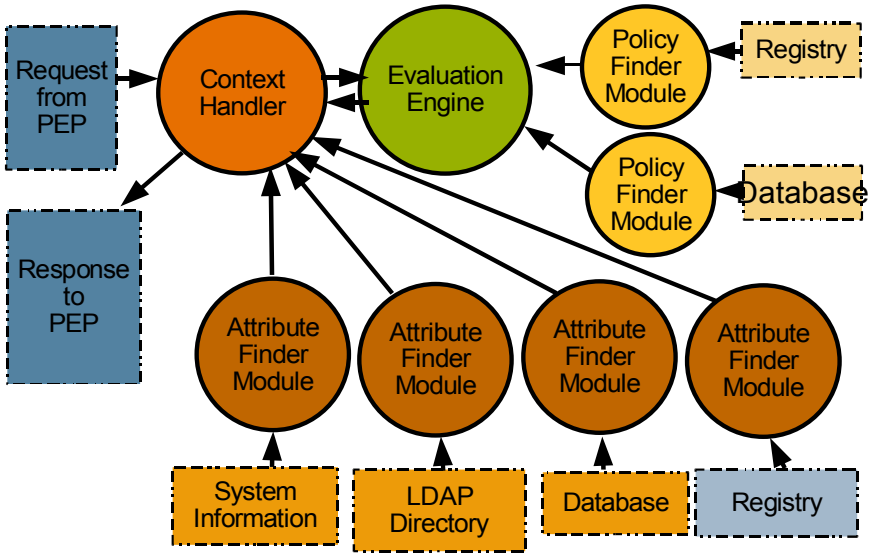
Just as multiple Rules can be combined in a Policy, multiple Policies can be combined in a PolicySet. PolicySets can also contain other PolicySets, allowing arbitrarily deep nesting. PolicySets and Policies can have <Target> elements just like Rules do – just as for Rules, these are simple constraint functions to make it efficient to determine whether the Policy or PolicySet applies to a specific access request. A PolicySet, including all its contained Policies and PolicySets, is evaluated only if its <Target> is "true" for a given request. Likewise, a Policy, and all its contained Rules, is evaluated only if its <Target> is "true" for a given request.

Just as Policies have "Rule Combining Algorithms", PolicySets have "Policy Combining Algorithms" to resolve differences in the results returned by its sub-ordinate policies.

A Policy or PolicySet may also have "Obligations". These assign values to variables that are returned along with the result – a policy may have one set of Obligations that are returned if its result is "Deny" and another set that are returned if its result if "Permit". For example, if result is "permit", a Policy may return an Obligation saying the accessed data must be destroyed within 30 days. The Policy Enforcement Point that invokes the XACML evaluation engine is responsible for carrying out Obligations.

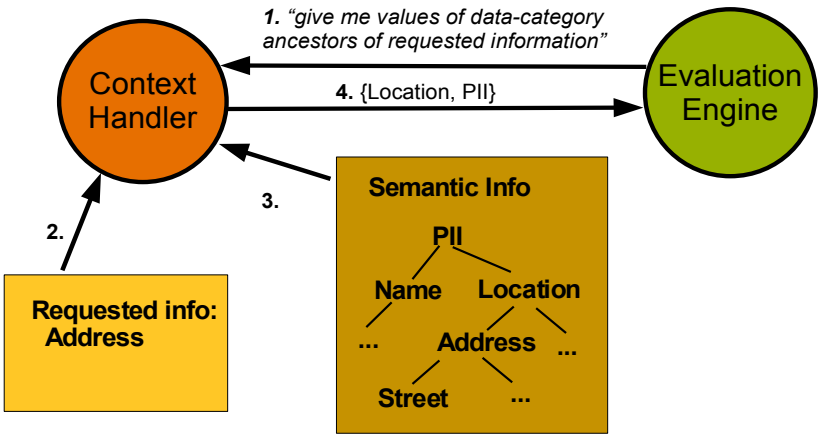
This overview has presented only the basic functionality of XACML. XACML also supports references to external policies, policy component definition and reference to avoid restating commonly used components, and many other features. One feature that may be relevant to privacy policies is the ability to place constraints on multiple subjects that may be involved in making a particular access request: for example, there may be the human user, the application being executed by the human user, the platform on which the application is running, the target destination for the accessed data, etc.

Applying Semantic Information to XACML Policies



In the XACML model, the Evaluation Engine requests all values for variables from the “Context Handler”. The Context Handler handles any format conversions, etc. that are required. The Context Handler is also the point at which semantic information can be associated with policies.

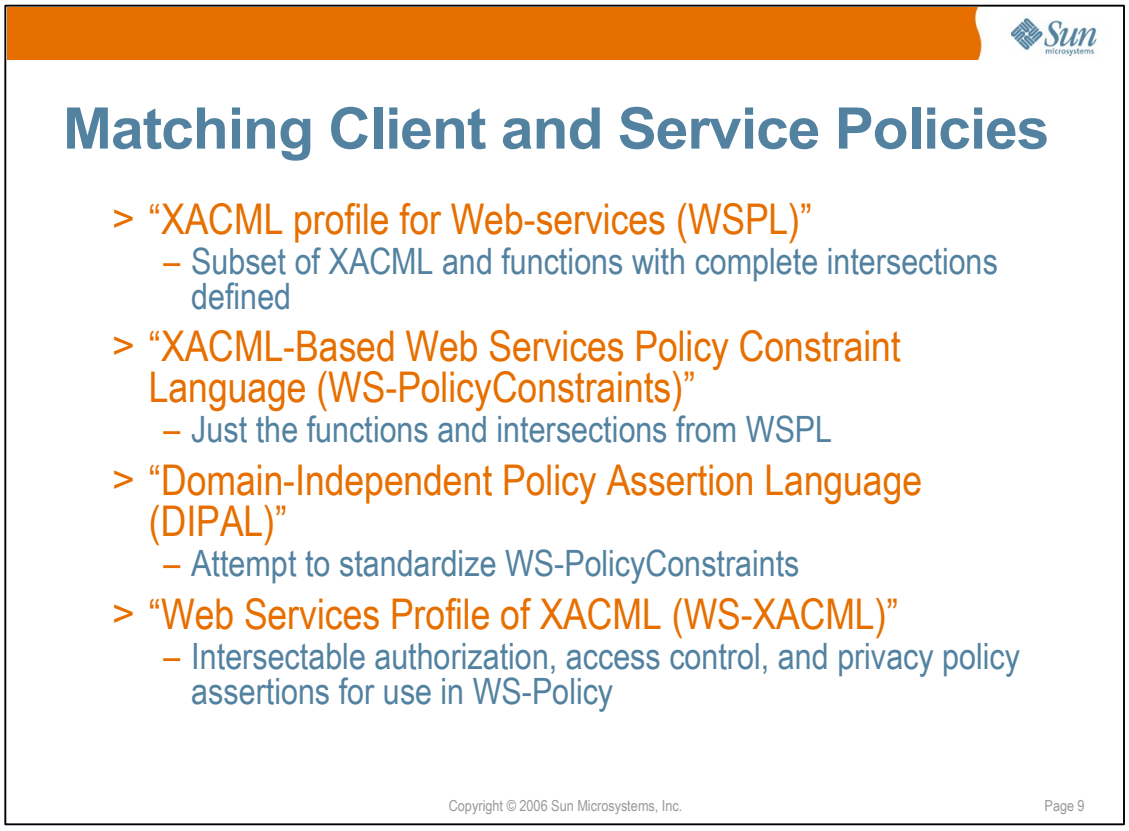
Applying Semantic Information to XACML Policies



An existing example from the "XACML Profile for Hierarchical Resources" is a standard AttributeId for "resource-ancestor-or-self", in which the Context Handler is assumed to have knowledge of the hierarchies to which each requested resource might belong, and returns the identity of the requested resource along with the identities of its ancestors in those hierarchies. As a specific example, if the requested resource is /Record/PersonalInfo/LocationInfo/Address, then this Attribute would resolve to { /Record, /Record/PersonalInfo, /Record/PersonalInfo/LocationInfo, /Record/PersonalInfo/LocationInfo/Address}. If a constraint says, ' "resource-ancestor-or-self is in /Record/PersonalInfo" ', then that constraint would be satisfied by this request.

In this example, the hierarchy is encoded into the identity of the resource, but that need not be the case. It is also possible to define a "*type*-ancestors-and-self" function that takes as input an AttributeId of DataType *type* and returns the value of that Attribute and the values of all its ancestors.

Integration of semantic information, such as RDF descriptions of categories of resources or subjects, into the Context Handler, is a useful avenue for extension of XACML. The Context Handler can also handle mappings of values used in a client's domain with equivalent values used in a service's domain.



Tim Moses of Entrust edited a Working Draft in the OASIS XACML TC called the "XACML profile for Web-services"; this was also referred to as the "Web Services Policy Language", or "WSPL". This was a subset the XACML syntax and functions that supported efficiently computing the intersection of any two policies, such as a client's policy and a service's policy. This Draft was eventually voted out-of-scope for XACML because it addressed all kinds of Web Services policies and also the association of policies with service endpoints; XACML's charter speaks only of "authorization, access control, and entitlement" policies.

I thought having a generic, intersectable policy language was a great idea, so I worked on ways to standardize this. Since WS-PolicyAttachment and WS-Policy had a lot of backing as ways to associate policies with endpoints and as the way to express Boolean combinations of policy assertions, I eliminated that functionality from WSPL. The result was "WS-PolicyConstraints", which defined the intersectable XACML functions as WS-Policy Assertions. We conducted a public discussion about standardizing this approach in OASIS at the beginning of this year under the name "Domain-Independent Policy Assertion Language" or "DIPAL", because we did not want to assume that "WS-PolicyConstraints" would be the language chosen. While there was a lot of interest in DIPAL, there were not enough big partners or resources committed to pursue standardization before the close of the public discussion period.

Hearing that XACML users wanted guidance on using XACML in a Web Services environment,, I took ideas from WS-PolicyConstraints and created a new "Web Services Profile of XACML" or "WS-XACML". This is the successor to WSPL as an XACML Working Draft. Among other things, it defines authorization, access control, and privacy assertions for use with WS-Policy that are based on XACML. It also defines how to compute the intersection of a client's assertion with a service's assertion, using the intersectable functions going back to the original WSPL.

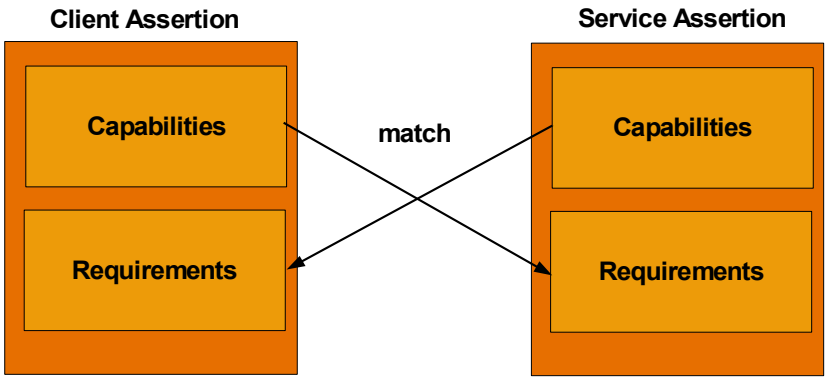
This approach offers a way to match client and service privacy policies as part of Web Service policies.

Matching Examples

- Client: x ≥ 30 Service: x = 10
  - Intersection empty – incompatible requirements/capabilities
- Client: x ≥ 30 Service: x = 35
  - Intersection: 35
  - This satisfies both policies
- Client: x in {A, B, C} Service: x in {B, C}
  - Intersection: x in {B, C}
  - Either value satisfies both policies
- All based on semantics of datatypes and arithmetic/sets
  - Usually a floor or ceiling operation or set intersection

These are some examples of how the intersections of constraints are computed. The defined intersections are very simple to compute (except for something like matching two regular expression functions), yet these types of computations can be done for a fairly rich set of comparison and set functions that allow the expression of rich constraints.

WS-XACML



Page 11

In WS-XACML, both parties may state a policy Assertion. Each party may have Capabilities and Requirements. Capabilities are information that the party is able and willing to provide in conjunction with having its Requirements met.

Capabilities may be expressed in the form of an XACML Request – a set of Attributes with values, or in the form of a set of constraints. Using constraints allows sets of providable values to be described.

Requirements may be expressed in the form of an XACML Policy or PolicySet, or in the form of a set of constraints.

Two Assertions are compatible if the Requirements of each are satisfied by the Capabilities of the other. Note that there must be a Capability that satisfies each Requirement, but that not all Capabilities might be needed to satisfy a particular set of Requirements.

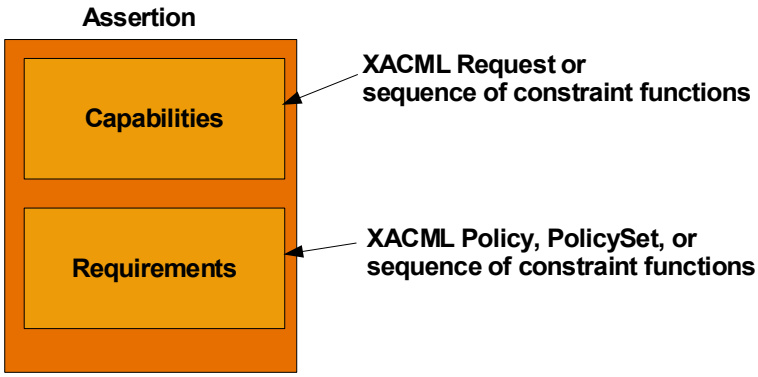
Obligations are represented as Requirements; the other party must have a corresponding Capability that indicates the party is able and willing to satisfy some obligation.

There may be different Assertions for different pieces of requested information, and there might even be multiple Assertions for the same piece of requested information: different combinations of Attributes that the party is able and willing to provide,

Negotiation protocols have not yet been discussed. Each party might make its Requirements known to the other party as a first step. The other party then determines whether its Capabilities can satisfy the Requirements, and if so, the minimal set. Each party might then send its minimal set of Capabilities to the other party.

So far, WS-XACML is defined only for determining policy compatibility, not for negotiating binding agreements. Providing a copy of the XACML Assertion with minimal Capabilities to the other party in a signed SAML Assertion with a validity period, however, might be a format for use in a binding agreement.

WS-XACML



All combinations can be matched, except for Requirements consisting of Policy or PolicySet and Capabilities consisting of non-equality constraint functions. But Policy, PolicySet really used only for very complex, non-normalized policies; more often in the envisioned use-cases, sequence of constraints, with possibly multiple alternative Assertions, should suffice.

References

[1] XACML References: Products and Deployments
http://docs.oasis-open.org/xacml/xacmlRefs.html#Products

[2] XACML Profile for Web Services (WS-XACML)
draft 5, 9 October 2006
http://www.oasis-open.org/committees/download.php/20643/xacml-3.0-profile-webservices-spec-v1.0-wd-5-en.pdf


Sun
microsystems
XACML-BASED
PRIVACY POLICY
LANGUAGES
Anne Anderson
anne.anderson@sun.com