OASIS eXtensible Access Control Markup Language (XACML)

Technical Committee

Issues List

Version 06

March 27, 2002

Ken Yagen, Editor

Purpose 4
Introduction 4
Use Case Issues 5
Group 1: Group Name 5
Design Issues 5
Group 1: Group Name 5
Policy Model Issues 5
Group 1: Rules 5
ISSUE:[PM-1-01: Negative Authorizations] 5
ISSUE:[PM-1-01-A: Implementing global deny and Meta-Policies] 6
ISSUE:[PM-1-02: Post-Conditions] 13
ISSUE:[PM-1-03: Post-Conditions as a term] 16
ISSUE:[PM-1-04:References to attributes in XACML predicates] 17
ISSUE:[PM-1-05: how NOT-APPLICABLE impacts a combinator expression] 18
ISSUE:[PM-1-06: result of <N-OF n=0> combinator expression] 21
ISSUE:[PM-1-07: How can the set of combinators be extended?] 22
ISSUE:[PM-1-08: syntax for <applicablePolicyReference>] 22
Group 2: Applicable Policy 23
ISSUE:[PM-2-01: Referencing Multiple Policies] 23
ISSUE:[PM-2-02: Target Specification] 24
ISSUE:[PM-2-03: Meaningful Actions] 25
ISSUE:[PM-2-04: Indexing Policy] 26
ISSUE:[PM-2-05: Ensuring Completeness] 26
ISSUE:[PM-2-06:Encapsulation of XACML policy (was Policy Security)] 27
ISSUE:[PM-2-07: valueRef type] 28
ISSUE:[PM-2-08: Outcome of policies and their combination] 28
Group 3: Policy Composition 29
ISSUE:[PM-3-01: Combining Policy Elements] 29
ISSUE:[PM-3-02: Specifying Policy Outcome] 30
ISSUE:[PM-3-03: multiple Base Policies] 30
ISSUE:[PM-3-03A: default PDP result] 31
ISSUE:[PM-3-04: Pseudo Code for Combiner Algorithms] 32
Group 4: Syntax 32
ISSUE:[PM-4-01: Triplet Syntax (was Syntactic Sugar)] 32
ISSUE:[PM-4-02: Policy names as URIs] 33
ISSUE:[PM-4-03: Required type in policy] 33
ISSUE:[PM-4-04:syntax extension] 33
ISSUE:[PM-4-05:Policy Name a URI] 34
ISSUE:[PM-4-06:Comment element] 34
ISSUE:[PM-4-07:policy element in a rule] 34
ISSUE:[PM-4-08:XML elements include xsi:type] 35
ISSUE:[PM-4-09:complex types] 35
ISSUE:[PM-4-10:preserve PAP identity] 35
Group 5: SAML Related 36
ISSUE:[PM-5-01: Non-SAML Input] 36
ISSUE:[PM-5-02: Wildcards on Resource Hierarchies] 36
ISSUE:[PM-5-03: Roles and Group Hierarchies] 37
ISSUE:[PM-5-04: SAML Assertions URI] 37
ISSUE:[PM-5-05: XPath] 38
ISSUE:[PM-5-06: Multiple actions in single request] 38
ISSUE:[PM-5-07: Delegation] 39
ISSUE:[PM-5-08: saml;Action is a "string"] 40

ISSUE:[PM-5-09: saml;AuthorizationQuery requires actions] *41*
ISSUE:[PM-5-10: single subject in AuthorizationQuery] *41*
ISSUE:[PM-5-11:XACML container in SAML] *41*
ISSUE:[PM-5-12:derive attribute from saml:AttributeValueType] *42*
ISSUE:[PM-5-13: Base Policy supplied as part of AuthorizationDecisionQuery] *42*
ISSUE:[PM-5-14: Resource Structure] *42*
ISSUE:[PM-5-15: Attribute reference tied to object] *43*
ISSUE:[PM-5-16: Arithmetic Operators] *43*
ISSUE:[PM-5-17: Boolean Expression of rules] *43*
Group 6: Predicate Cononicalization *44*
ISSUE:[PM-6-01: SAML Assertions URI] *44*
Group 7: Extensibility *44*
ISSUE:[PM-7-01: XACML extensions] *44*
Group 8: Post Conditions *45*
This group was created out of issues raised in Michiharu's proposal for post conditions. See Also Issues PM-1-02 and PM-1-03 for more on post conditions *45*
ISSUE:[PM-8-01:] (4.1) Internal v.s. external post conditions *45*
ISSUE:[PM-8-02:] (4.2) Mandatory v.s. advisory post conditions. *45*
ISSUE:[PM-8-03:] (4.3) Inapplicable *45*
ISSUE:[PM-8-04:] (4.4) Base policy v.s. policy reference *46*
ISSUE:[PM-8-05:] (4.5) How to return post conditions via SAML *46*
ISSUE:[PM-8-06:] (4.6) When to execute post condition *49*
ISSUE:[PM-8-07:] (4.7) Extension point *49*
MISCELLANEOUS ISSUES 50
Group 1: Glossary *50*
ISSUE:[MI-1-01: Consistency] *50*
ISSUE:[MI-1-02: Definition of Policy vs. Rule] *50*
ISSUE:[MI-1-03: Definition and purpose of Target] *51*
Group 2: Conformance *52*
ISSUE:[MI-2-01: Successfully Using] *52*
Group 3: Patents, IP *53*
ISSUE:[MI-3-01: XrML] *53*
Group 4: Other Standards *54*
ISSUE:[MI-4-01: RuleML] *54*
ISSUE:[MI-4-02: RAD] *55*
ISSUE:[MI-4-03: DSML] *55*
ISSUE:[MI-4-04: Java Security Model] *56*
DOCUMENT HISTORY 56

Purpose

This document catalogs issues for the eXtensible Access Control Markup Language (XACML) developed the Oasis eXtensible Access Control Markup Language Technical Committee.

Introduction

The issues list presented here documents issues brought up in response to draft documents as well as other issues mentioned on the xacml mailing list, in conference calls, and in other venues. The structure of this document was taken from the Security Assertion Markup Language (SAML) Issues List document maintained at the Security Services Technical Committee document repository. Each issue is formatted as follows:

ISSUE:[Document/Section Abbreviation-Issue Number: Short name] Issue long description. Possible resolutions, with optional editor resolution Decision

The issues are informally grouped according to general areas of concern. For this document, the "Issue Number" is given as "#-##", where the first number is the number of the issue group.

To make reading this document easier, the following convention has been adopted for shading sections in various colors.

Gray is used to indicate issues that were previously closed.

Blue is used to indicate issues that have been flagged as ready to close in the most recent revision. These require review and voting by the committee and they can be closed.

Yellow is used to indicated issues which have recently been created or modified or are actively being debated.

Other open issues are not marked, i.e. left white.

Issues with lengthy write-ups, that have been closed "for some time" will be removed from this document, in order to reduce its overall size. The headings, a short description and resolution will be retained. All vote summaries from closed issues will also been removed.

Use Case Issues

Group 1: Group Name

Design Issues

Group 1: Group Name

Policy Model Issues

Group 1: Rules

ISSUE:[PM-1-01: Negative Authorizations]

Authorizations can be either positive (permit) or negative (deny). Should we allow both?

See also PM-1-01-A which was split off from this issue.

Potential Resolutions:

[Michiharu] There seems to be agreement on the fact that the core schema should support positive authorizations only. Negative ones are supported as an extension.

[Tim] XACML shall address the requirement for "negative rules" by means of an "and-not-or" construct. [PM-1-01]

[Tim] We use a construct of the following form …

```
<and>
  <rule1/><rule2/><rule3/>
  <not>
    <or>
      <rule4/><rule5/>
</or></not></and>
```

Rule4 and rule5 specify circumstances under which, if either were to hold, access is to be denied. While rule1, rule 2 and rule3 specify circumstances, all of which must hold if access is to be granted.

Proposed Resolution:

XACML allows policy writers to specify positive (permit) or negative (deny) authorization. The negative authorization is specified using the effect element with "deny" in the rule with corresponding rule set combiner such as "meta-policy-1" meaning the global-deny semantics.

Using the rule combiner (XACML extension point), the semantics of the negative authorization varies depending on the user-defined rule combiner. PM-1-01-A discusses about the global-deny semantics.

Champion: Michiharu

Status: Closed

ISSUE:[PM-1-01-A: Implementing global deny and Meta-Policies]

Implementing global "deny" semantics using schema 0.8 and meta-policies

[Anne] USE CASE: policy is to deny access to Principal "Anne Anderson" under all conditions. The policy is distributed across many sub-policies, which are all combined to produce the global policy that is to be applied.

Michiharu's concern was with needing to put something like

```
<not><equal>
  <valueRef entity="principal">saml:Subject/NameIdentifier/Name</valueRef>
    <value>"Anne Anderson"</value>
</equal></not>
```

Into every sub-policy if there was no global "deny" syntax.

My proposed solution depends on the idea of having meta-policies. I think meta-policies solve multiple problems:

1. "Where do I get policies",

2. Knowing when you have obtained all the relevant policies,

3. Knowing how to combine policies

4. being able to implement global "deny" and meta-policies does not introduce any new syntax. It is just very explicit in specifying what "applicable policy" means.

Potential Resolutions:

[Anne] Each PDP (or PRP) needs to be configured with a single policy that serves as that PDP's "meta-policy". The syntax of this single policy is exactly that in 0.8.

This "meta-policy" determines where and under what conditions various sub-policies are retrieved. I may not be using <externalFunction> correctly, or the subpolicies may need more enclosing namespace information, but I hope these examples will give the idea. The final example shows how global "deny" semantics are implemented.

EXAMPLE SIMPLE META-POLICY FOR DISTRIBUTED POLICIES:

```
 <?xml version="1.0" encoding="UTF-8"?>
 <applicablePolicy xmlns=...  issuer="<identity that ultimately controls policy for this PDP>"
policyName="...">
   <!-- target omitted, since this policy applies to all targets -->
   <policy>
     <and>
       <externalFunction>http://www.site1/policy1.xml</externalFunction>
       <externalFunction>http://www.site2/policy2.xml</externalFunction>
       ...
     </and>
   </policy>
 </applicablePolicy>
```

What is found at each of the <externalFunction> locations is another <applicablePolicy>, which may be more specific as to which resources it applies to (that applicablePolicy in turn may refer to still other policies). If one of these <applicablePolicy> elements does not apply to the current request, then the result is "does not apply" and does not affect the result of the <and> evaluation.

META-POLICY THAT USES SUB-POLICIES BASED ON RESOURCE

```
 <?xml version="1.0" encoding="UTF-8"?>
 <applicablePolicy   xmlns=...    issuer="<identity that ultimately controls policy for this PDP>"
  policyName="...">
   <!-- target omitted, since this policy applies to all targets -->
   <policy>
     <or>
       <and>
         <equal>
           <valueRef>saml:Resource</valueRef>
           <value>"file:/host1/*"</value>
         </equal>
         <externalFunction>http://www.site1/policy1.xml</externalFunction>
       </and>
       <and>
         <equal>
           <valueRef>saml:Resource</valueRef>
           <value>"file:/host2/*"</value>
         </equal>
         <externalFunction>http://www.site2/policy2.xml</externalFunction>
       </and>
       ...
    </or>
   </policy>
 </applicablePolicy>
```

META-POLICY THAT IMPLEMENTS GLOBAL DENY SEMANTICS

```
  <?xml version="1.0" encoding="UTF-8"?>

  <applicablePolicy   xmlns=...    issuer="<identity that ultimately controls policy for this PDP>"
   policyName="...">
    <!-- target omitted, since this policy applies to all targets -->
    <policy>
      <and>
        <not>
          <equal>
            <valueRef entity="principal">saml:Subject/NameIdentifier/Name</valueRef>
            <value>"Anne Anderson"</value>
          </equal>
        </not>
        <or>
          <and>
            <equal>
              <valueRef>saml:Resource</valueRef>
              <value>"file:/host1/*"</value>
            </equal>
            <externalFunction>http://www.site1/policy1.xml</externalFunction>
          </and>
          <and>
            <equal>
              <valueRef>saml:Resource</valueRef>
              <value>"file:/host2/*"</value>
            </equal>
            <externalFunction>http://www.site2/policy2.xml</externalFunction>
          </and>
          ...
        </or>
      </and>
    </policy>
  </applicablePolicy>
```

For administrative ease in a more realistic situation, the set of globally denied attribute/value combinations would be placed in one <externalFunction> policy.

[Ernesto] I support this proposal. I believe it could deal smoothly with the distributed scenario Anne described many times during the last conference call. It goes in the same direction of a previous suggestion of mine (deal with composition and distributed deployment at the ApplicablePolicy level), but does it far better. However, I would suggest some minor observations/amendments (otherwise there is no fun :-))

1. Maybe this is trivial, but any change to the current schema should keep policies fully embeddable in the Applicable policy element, besides being able to point to them using external functions. In simple environments there will be only one local policy, stated in a single document.

2. I happen not to like very much using the word "meta-policy" to describe this proposal, for several reasons some of which would be too long to explain in this message. Basically, I regard Anne's technique mainly as a way to define how a global policy can be deployed in distributed, independently maintained retrieval units. In passing, it also solves the problem of stating which criterion should be applied to compose the outcome of such units (this is essential when "deny" is a possible outcome, as the criterion may have an impact on what actually needs to be retrieved), but I cannot convince myself this requirement is equally important. I believe (but would like to hear the opinion of the industrial researchers on this one) that there will be a default policy composition technique that will be used 99.9% of the times. Therefore, in the schema I would prefer to concentrate the deployment description functionality in a new element, perhaps called "ApplicablePolicies" , possibly defined as an extension of the base (Applicable)Policy type. This element could optionally (via an attribute) specify the composition criterion as well. Tim, what are your views?

[Hal] I am not sure if I agree with Anne's approach. I certainly like it better than the alternative proposed. I actually thought we had previously agreed that there had to be some rules (policy) for determining how independently created policies should be combined to achieve an authorization decision.

Instead of meta-policy, which I think Ernesto fears will be take to mean "more abstract policy" or "policy about policy", perhaps something like Policy Federation Rules would be better.

It seems to me the key issues are:

1. Where and how are PFR specified? Anne's approach is a distinct XML document, which must be consistent throughout the policy federation. This seems reasonable to me.

2. What are the possible PFR's? I think "AND" is impractical, and "OR" is most likely, however some kind of best-match-to-target is conceivable although perhaps too expensive to implement in practice.

3. Do all legal PFR's have to support all decision strategies? I have been thinking about this and I think the right approach is to explicitly call out the possible decision strategies and for each legal PFR state which can or cannot be used.

Here's what I have so far on decision strategies.

Strategy I - Basic

1. Collect all applicable policies
2. Obtain all required inputs
3. Evaluate all policies
4. Apply PFR to resolve conflicting results

Strategy II - Optimized

1. Collect all applicable policies
2. Use PFR to create equivalent combined policy
3. Evaluate policies incrementally, gathering inputs as needed, defer evaluations based on inputs requirements (this for example allows "lazy authentication" where authentication is not done if the result can be determined without it)
4. Once the result is known, stop evaluation

Strategy III- Incremental collection

1. Collect "some" policies
2. Obtain required inputs
3. Evaluate current policy set
4. Use PFR to combine latest results with previous results (if any)
5. If result is known, stop evaluation
6. If not all policies have been collected, repeat previous steps

These are all the possibilities I can think of. Can anyone think of others? I think anything proposed to date works equally for I and II, but not all work for III. However, we may find future possibilities that only work for one of them.

To answer Ernesto's question, our product uses "OR" for authorization decisions and "AND" for audit decisions and there have been no complaints. However we do not have post conditions, which may change things.

As far as the global deny, I would like to understand the requirements better. It seems the problem Anne is trying to solve is "master policy admin can globally deny regardless of what the policy combining rules are."

Is this the right problem to solve? If an "OR" combining rule is used (which I happen to think is the most common case) then any admin can implement a global deny without any special machinery. I think the example given is a red herring to some extent, because the right way to cut off an individual user is to change their attributes at the Attribute Authority or revoke their credentials.

The problem I see is that most evaluation engines will want to use a relatively fixed decision strategy in order to optimize it according to the criteria that apply in that environment. Finding it out in the middle of policy evaluation will interfere with this goal.

[Michiharu] I also support Anne's proposal. I think this technique deal with the distributed scenario nicely. I said the similar idea that uses an external function to call sub applicable policies in the policy model con-call on Dec. 17 but Anne's description is much more concrete and easy to understand. For the global deny policy, I agree that this technique is useful to specify the global deny semantics. If this technique is agreed, we may need more intuitive name for the externalFunction.

[Pierangela] I agree with the fact that the current proposal is able to implement the global deny scenario. No doubt about that: if you restrictions (i.e., the deny you want to enforce) ANDED with the other possible policies nobody will be able to overrule your restrictions.

The reason why I am not too excited with the current proposal is that it seems perfectly fine for communicating policies, but it seems complex to manage.

First of all you have to make sure that the applicable policy is in a single place (sure possibly using URL of other policies) but you cannot allow overlapping targets (which seemed to be the case till now, I believe).

Second the priority of your rules is explicitly managed with the policy definition, which may make administration heavy. Who is in charge of specifying the applicable policy? This will be the only one able to specify global deny: if understand Tim/Anne's proposals correctly possible negative authorizations in other policies have the effect only within that policy (this is fine with me, it seems conceptually clean).

Now for instance, suppose you want to enforce a situation in which any of us can grant authorizations and, possibly denials, for some access and a denial-take-precedence policy should be enforced (meaning it sufficient that one of us says "deny (because of a negative authorization), and the access should be rejected. How do you enforce this? You cannot have the different administrators operate on the applicable policy (meaning actually have writing privilege on that document).

[From 2/18 minutes] A metapolicy can state how you should combine classes of rules or of policies. For instance, it could query attributes of rules (e.g., sign) or of policies (corporate policies as opposed to department policies). Simon notes there are two components. one is how to solve conflicts, you do not really need this syntax. The other level is when you start combining policies, here you need the expressive power of the metapolicy language. So for meta-policies associated with elementary policies we could have a pre-defined URI expressing the conflict resolution policy without need to use the metapolicy specification language. It is however noted that at the URI you should find a metapolicy expressed.

NOTE: We once said it would be nice if we had at least an example of meta-policy in our proposal. Should we have it explicitly mentions ``meta-policy one"?

Proposed Resolution:

the syntax for <rule> allows for the <rule> to return an <effect> of "permit" or "deny". It is up to the combiner in the <policyStatement> that uses a <rule> to determine the effect of a <rule> that returns "deny". Likewise, it is up to the combiner in the <policyCombinationStatement> that uses a <policyStatement> to determine the effect of a <policyStatement> that returns "deny".

The following example combiners can be used to implement "global deny" semantics for a <rule>. Since an "indeterminate" rule might have evaluated to "deny" if sufficient information had been supplied, these examples treat "indeterminate" results like "deny".

GLOBAL DENY RULE COMBINER:

```
for <rule> in <ruleSet> {
  boolean atLeastOnePermit = false;
  effect = eval(<rule>);
  if (effect == "deny" || effect == "indeterminate") {
     return "deny";
  } else if (effect == "permit") {
     atLeastOnePermit = true;
  }
}
if (atLeastOnePermit) {
  return "permit";
} else {
  return "not applicable";
}
```

GLOBAL DENY POLICY COMBINER:

```
for <policy> in <policySet> {
  boolean atLeastOnePermit = false;
  effect = eval(<policy>);
  if (effect == "deny" || effect == "indeterminate") {
    return "deny";
  } else if (effect == "permit") {
    atLeastOnePermit = true;
  }
}
if (atLeastOnePermit) {
  return "permit";
} else {
  return "not applicable";
```

}

Policy and policy combination writers that do not wish to support "global deny" semantics can specify different combiners.

Policy combination writers should publish the combiner they use to policy writers so that consistent semantics are maintained: if a policy combination writer is implementing "global deny", then the policy writers should be aware that returning an effect of "deny" will by itself result in denial of access.

Champion: Anne

Status: Closed

ISSUE:[PM-1-02: Post-Conditions]

The current schema [Tim, Jan.3] mentions post-conditions, distinguishing between external and internal, depending on whether their execution requires dialoging with external entities. The current schema suggests (via a comment) that post-conditions can be expressed as invocations of SOAP services. Post-conditions are still to be discussed in details: what is their semantics; how are they executed? A complication of post-conditions associated with a rule involves the distributed scenario (see POLICY COMPOSITION issue). In fact, if I say that a post-condition should be applied whenever a rule fires then I have to evaluate *all* rules. A possible way to overcome this problem is to consider that post-conditions associated with the authorizations that were evaluated to get to an access decision should be executed [Tim]. Note: a possible drawback of this approach is that deterministic behavior may be lost. For instance, there may be N rules applying to an access. If the evaluation of 1 of them brings to a ``permit" decision (so there is no need to evaluate the others). Then, you would ignore the post conditions possibly associated with the other N-1. Different execution of the same request on the same state could then have a different behavior (because a different rule is considered as authorizing the request.

[Tim] The alternative view is that post-conditions must be executed if and only if the associated rule contributes to the permit decision.

[Polar] What is the purpose for actions (i.e. these post conditions) after checking a policy? What types of actions are allowed? Do they change the state of the policy?

[Pierangela] examples that were brought up for post-conditions were things like "logging the request", essentially they are actions that the system executes in response to granting an access, or simply having evaluated the authorizations (discussion on the specific behavior is still open).

Do they change the state of the policy? If you mean the set of rules I guess the answer is no (they should not change the rules). But again, post-conditions are one of the issues which have not discussed fully.

[Polar] Well, I had originally thought that a "post-condition" would be something that would be true if the policy evaluated to true according to its input. That is, a "post-condition" should be a logical consequence, but maybe not fully derivable by all available information. This post-condition would merely be some advice to the evaluator.

Such as Policy stating that:

Subject is in Role of MissleLauncher to the Resource of Missile on Action Launch.

Post-condition Subject is dangerous.

I really don't like the fact that these post conditions mandate that some generic operation be performed, i.e. it could be used to alter state, especially the state of the policy.

[Simon] Post-condition is executed after the rule fires and does not affect grant/deny

Outcome of the rule. With this definition we can not predict which post condition(s) will be executed for a given authorization request. This is not desirable. One way to make post-conditions predictable is to associate post condition not with a rule but with the outcome of grant or deny, e.g.:

on_grant do_something
on_deny do_something

That means every time any subject is granted (or denied) action on any resource all post-conditions listed in on_grant (or on_deny) will be predictably executed. On_grant and on_deny post-conditions could be associated with specific action, subject, and resource triplet, meaning that given post-condition will be executed every time subject is granted or denied permission to access resource.

on_grant(action, subject, resource) do_something;
on_deny(action, subject, resource) do_something;

[John]

> Post-condition is executed after the rule fires and does not affect
> grant/deny outcome of the rule.

I thought this was only true of *external* post-conditions? I thought that an internal post-condition must be executed (by the PDP) BEFORE the response is asserted, and therefore does affect the outcome...

The spec says:

"...Post-condition - A process specified in a rule that must be completed in conjunction with access. There are two types of post-condition: an internal post-condition must be executed by the PDP prior to the issuance of a "permit" response, and an external post-condition must be executed by the PEP prior to permitting access..."

I'm assuming that the "musts" here imply that the required actions are successfully executed. Is this not the case?

[Simon] The way I remember post-conditions discussions is that outcome of internal post condition does not affect the outcome of azn decision, i.e., first grant (or deny) is computed and then internal post-condition is executed. If, for example, pdp fails to add a record to the log it still returns computed outcome (grant or deny) to the pep. So the internal post-condition may not be successfully executed by the pdp.

[Tim] This can be accomplished with the current syntax.

applicablePolicy/policy/rule+post-condition

This post-condition is executed if access is permitted.

applicablePolicy/policy/not/Rule+post-condition

This post-condition is executed if access is denied.

[Bill]

If given this:

> With this definition we can not predict which post condition(s) will be

> executed for a given

> Authorization request. This is not desirable.

'do_something' cannot be guaranteed:

> on_grant(action, subject, resource) do_something;

> on_deny(action, subject, resource) do_something;

Because that would require acknowledgement that it occurred (implying dependence on grant/deny). Sounds like 'post condition' in this sense is more like 'post request'.

[Hal] I clearly remember that the sense of the group was that the PDP MUST insures that an internal post condition occurs, but not necessarily before the permit decision is returned. Post conditions were never considered optional. They are just as required for "permit" as pre-conditions are. That was the rationale for the name.

Potential Resolutions:

[Tim] XACML shall require the PDP/PEP to execute just those post-conditions that accompany the rules that contribute to the "permit" decision. [PM-1-02]

See email to list from Michiharu on 2/11/2002 with a proposal for post conditions

Proposed Resolution:

[From Michiharu and Anne]

[We use the term "obligation" to mean what we have previously been calling "post condition". The issue of the term is addressed in PM-1-03.]

Obligations are annotations that MAY be specified in a policyStatement and/or policyCombinationStatement that should be returned in conjunction with an authorization decision meaning that the obligations(s) SHOULD be executed by the PEP. The obligation is specified using URI reference with optional arguments. The actual meaning of each obligation depends on the application. It also depends on the configuration of the PEP and/or PDP. If the PEP does not recognize an obligation, the PEP should deny access.

The set of obligations returned by each level of evaluation includes only those obligations returned by rules, policyStatements, or policyCombinationStatements that were actually evaluated by the combiner algorithm, and associated with the effect element being returned by the given level of evaluation. For example, a policy set may include some policies that return Permit and other policies that return Deny for a given request evaluation. If the policy combiner returns a result of Permit, then only those obligations associated with the policies that were evaluated, and that returned Permit are returned to the next higher level of evaluation. If the PDP's evaluation is viewed as a tree of policyCombinationStatements, policyStatements, and rules, each of which returns "Permit" or "Deny", then the set of obligations returned by the PDP will include only the obligations associated with evaluated paths where the effect at each level of evaluation is the same as the effect being returned by the PDP.

Champion: Simon

Status: Closed

ISSUE:[PM-1-03: Post-Conditions as a term]

[Bill] I know that it is late to bring this up, but I find the term 'post condition' unintuitive. Typically, this phrase means the *state* of something after an action, not something to be acted upon. It seems that the way we are using the term implies quite a bit about the context of what is being done. (post what? where?) I think this is being demonstrated by the discussions surrounding the scope of said phrase. In my mind, it would seem that something like 'adjunct policy' or 'adjunct policy condition' would be more appropriate?

[Pierangela] I share this feeling (incidentally, I brought it up in the last conference call, and also in previous once). I was interpreting them more as "actions" than "conditions".

[Pierangela] in today's TC conference call, some people mentioned that "action" is already used with different semantics (=the operation the principal is requesting). That's true, so we should find another term. The point is, however, that the semantics of "post conditions" now seems really to be a reaction of the system, not the evaluation of a state, so terminology should reflect the semantics.

Potential Resolutions:

1. adjunct policy
2. adjunct policy condition
3. actions

Bill: for me, one of the problems with the term 'post-condition' is that it technically refers to the state* of something after an event, not something that must be done (as is the case with the term 'pre-condition'). this can become confusing when working in other contexts (like UML: Postconditions - Describe the state of the system, and perhaps the actors, after the use case is complete...")

for starters, how about these?

Stipulation, provision, proviso, constraint, obligation, caveat, directive, regulation

i am sure we can come with a number of alternative terms that will work. Personally, I like 'obligation', because in this model this is really what you have: the PEP has an obligation to enforce the rulings of the PDP (i.e. GRANT) under the terms defined by the PDP (e.g. 'delete after 30 days') -- if it cannot it must DENY.

Proposed Resolution:

At the March, 2002 Face-to-Face meeting, we agreed to use the term "obligation" to express an annotation associated with an access decision that is returned to a PEP. This term replaces our former use of "post-condition".

Champion: Bill

Status: Closed

ISSUE:[PM-1-04:References to attributes in XACML predicates]

What information needs to be provided in order to refer to an attribute in an XACML policy predicate?

Potential Resolutions:

Proposed Resolution:

References to attributes associated with the access request in XACML predicates consist of a URI to a document instance that contains the value of the attribute to be evaluated, a URI for the schema for the document, a schema-dependent path for locating a particular attribute instance in the document according to the schema, and an optional name for the Attribute Authority trusted to assign values for this attribute. The AA is located using the PKI with which the PDP is configured.

Vote:

2/21: There was considerable discussion about whether this was ready to close. The feeling was that we needed to see a specific proposal either free standing or in the working spec before we could vote to close. The issue was raised as to whether we should use XPath expressions here. It was not closed

Champion: Anne

Status: Open

ISSUE:[PM-1-05: how NOT-APPLICABLE impacts a combinator expression]

A "combinator expression" is a combination of predicates, where possible combinators are <AND>, <OR>, <NOT>, <N-OF>, <ORDERED-[AND|OR|N-OF]>. This list of Combinators can be extended.

Example:

```
<AND>
  predicate1,
  predicate2,
  predicate3
</AND>
```

The issue occurs when one or more of the predicates in the list returns a result of NOT-APPLICABLE (this can occur if the predicate is a <referencedPolicy>). What should the result of the combinator expression be? What if ALL predicates in the combinator expression return NOT-APPLICABLE?

Potential Resolution:

[Anne]

a) Any predicate evaluating to NOT-APPLICABLE is logically removed from the combinator expression.

Example: if predicate3 in the example above returned a result of NOT-APPLICABLE, then the combinator expression is the result of

```
   <AND>
     predicate1,
    predicate2
  <AND>
```

b) An empty combinator expression has the following results:

```
<AND></AND>   -> TRUE
<OR></OR>     -> FALSE
<NOT></NOT>   -> TRUE
<N-OF></N-OF> -> FALSE
```

<ORDERED-[whatever]> has same result as [whatever] above. Extended combinators must define the result of an empty expression.

Example: If predicates 1, 2, and 3 in the example above all evaluate to NOT-APPLICABLE, then the combinator expression is <AND></AND>, and the result is TRUE.

b)-alternative: An empty combinator expression has a result of NOT-APPLICABLE.

[Polar] It's sort of like Anne's alternative #2 below with a couple of differences.

First, NOT-APPLICABLE (or Inapplicable?) and Error, are values that do not have an XML representation and are merely a artifact of evaluating policy expressions.

I propose the following consistent semantic model.

T = true, F = false, N = NOT-APPLICABLE, E = Error

The basic crux is that getting a NOT-APPLICABLE in the equation is as if its the NOT-APPLICABLE value isn't even there. For instance,

```
    (and  x N y) = (and x y)
    (or   x N y) = (or x y)
```

I think that is the semantics we want. That is to say, if the policy doesn't apply, it doesn't enter into the equation. I also surmise to keep things easily consistent in inductive arguments about ANDs and ORs of sequences. The AND or OR of a zero length sequence of values can be anything constant we want, but the minimum element NOT-APPLICABLE would make the most sense, since (and x N) = (and x), from our assumption above, and, (and x) = x, which is still another wily assumption, but makes sense,

So therefore (and N) = N, but from above, (and N) = (and), Therefore, (and) = N

So we would have,

```
  <and></and> = NOT-APPLICABLE
  <or></or>   = NOT-APPLICABLE
```

Also, to satisfy Hals "the customer's want it", I am almost on the side of allowing NOT in the language with the following semantics:

p	NOT p
T	F
F	T
N	N
E	E

That is to say NOT of NOT-APPLICABLE is still NOT-APPLICABLE. Then NOT distributes through the AND and ORs (i.e. DeMorgan's Law) quite nicely.

```
(NOT (AND N x)) = (OR (NOT N) (NOT x))
 (NOT x)        = (OR N (NOT x))
 (NOT x)        = (NOT x)
```

```
(NOT (OR N x))  = (AND (NOT N) (NOT x))
  (NOT x)       = (AND N (NOT x))
  (NOT x)       = (NOT x)
```

However, differing from alternative #2 in the proposal below, I believe <NOT></NOT> shouldn't exist, and it should have one and only one constituent. And empty NOT is a syntax error, as well as having more than one, i.e. <NOT> x y </NOT> shouldn't type check either. (how do you say that in XML? minoccurs=1, maxoccurs=1?).

For completeness the truth tables in the 4-valued logic are below for "and", "or" and "not", (ed note: truth tables left out. See original email)

Proposed Resolution:

A <rule> will return NOT-APPLICABLE under the following conditions:

<rule> Truth Table:

Target	Condition	Effect
match	match	[Effect]
match	no-match	Inapplicable
match	Indet.	Indet.
no-match	match	Inapplicable
no-match	no-match	Inapplicable
no-match	Indet.	Inapplicable

It is up to the combiner in the <policyStatement> that uses a <rule> to determine the effect of a <rule> that returns "Inapplicable". Likewise, it is up to the combiner in the <policyCombinationStatement> that uses a <policyStatement> to determine the effect of a <policyStatement> that returns "Inapplicable".

The example "GLOBAL DENY" combiners proposed in PM-1-01A can be used to implement "remove inapplicable elements from the computation" semantics.

The following example combiners can be used to implement "inapplicable same as deny" semantics. Such semantics might be desired where all rules are intended to be applicable, so a result of inapplicable indicates some breakdown in the consistency of the system.

INAPPLICABLE GLOBAL DENY RULE COMBINER:

```
if (<ruleSet> == null) {
  return "deny";
}
for <rule> in <ruleSet> {
  effect = eval(<rule>);
  if (effect == "deny" ||
      effect == "indeterminate" ||
      effect == "inapplicable") {
     return "deny";
}
return "permit";
```

INAPPLICABLE GLOBAL DENY POLICY COMBINER:

```
if (<policySet> == null) {
  return "deny"
}
for <policy> in <policySet> {
  effect = eval(<policy>);
  if (effect == "deny" ||
      effect == "indeterminate" ||
      effect == "inapplicable") {
    return "deny";
}
return "permit";
```

Champion: Anne

Status: Closed

ISSUE:[PM-1-06: result of <N-OF n=0> combinator expression]

We all agreed that <N-OF n=[something greater than 0]> was an error if there were not at least n predicates to be evaluated. We also agreed that the semantics of <N-OF> were "at least n of". We did not agree on what should be the result of <N-OF n=0>.

Potential Resolution:

<N-OF n=0> results in TRUE, regardless of the results of the predicates in the combinator expression.

Champion: Anne

Status: Open

ISSUE:[PM-1-07: How can the set of combinators be extended?]

We agreed at the March, 2002 F2F that XACML would define the <AND>, <OR>, <NOT>, <N-OF>, and <ORDERED-[AND|OR|NOT|N-OF]> combinators. How can a policy writer extend this set to define a new combinator, such as BEST-MATCH-OR?

Potential Resolution:

The set of Combinators may be extended by specifying a name for the new Combinator, a URI that is associated with the semantics of the new Combinator, and a type that specifies the way the URI is to be interpreted. Not all XACML PDPs will be able to interpret all extensions, but any PDP that can handle the specified type and can access the specified URI can handle the specified extended Combinator.

An example of a possible extended Combinator is BEST-MATCH-OR. The type for such an extended Combinator might be "JavaClass". The URI for each might point to a Java class that takes a set of Predicates as input and implements the semantics of the combinator to return a result of TRUE, FALSE, NOT-APPLICABLE, or ERROR.]

Proposed Resolution:

The combiner algorithm to be used by a given <policyStatement> or <policyCombinationStatement> is specified using a URI. XACML will specify a small set of mandatory-to-implement combiner algorithms. The algorithm associated with the URI MAY be descriptive text. Users are free to define other algorithms, although not all XACML-compliant PDPs will be able to apply them.

Champion: Anne

Status: Closed

ISSUE:[PM-1-08: syntax for <applicablePolicyReference>]

If a predicate in XACML references an <xacml:applicablePolicy>, what should the syntax for this reference be?

Potential Resolution:

The syntax should include a URI for <xacml:applicablePolicy> and a URI for the Policy Authority trusted to issue and sign this <xacml:applicablePolicy>. The name attribute in the referenced <xacml:applicablePolicy> must match the URI in the <applicablePolicyReference>. A chain of <applicablePolicyReference> that contains a cycle has a result of ERROR.

Champion: Anne

Status: Open

Group 2: Applicable Policy

ISSUE:[PM-2-01: Referencing Multiple Policies]

According to the current schema an Applicable Policy seems to refer to a single Policy. The discussions in the last conference call seem to assume that an Applicable Policy can refer to several Policies (distributed scenario and multiple issuers [Anne]). Is there agreement on this point? If so, the schema should be modified accordingly.

Group 1 issues are captured within this

[Tim] The current schema allows one possible way of achieving this. Separate applicable policies from independent PAPs (Policy Administration Points) may be combined in a single "applicable policy" by a PRP. This approach does, however, make the original PAPs anonymous.

Potential Resolutions:

[Tim] An XACML "applicable policy" will not reference external "applicable policies". However, it may "incorporate" external "applicable policies". [PM-2-01] [PM-3-01] [PM-5-03]

[Tim] An XACML "applicable policy" shall be capable of referencing an external "applicable policy", providing explicit rules for combining such policies. [PM-2-01] [PM-3-01] [PM-5-03]

Proposed Resolution:

Multiple policies may be referenced and combined using a <policyCombinationStatement>. This has the following syntax:

```
<policyCombinationStatement>
  <target/>
  <policySet Combiner="myURI">
    <policyDesignator>
      <policyRef> or <policyStatement> or
        <policyCombinationRef> or <policyCombinationStatement> or
        <saml:assertion>
      <policyMetadata>
    </policyDesignator>
    <policyDesignator>...</policyDesignator>
    <obligations />   OPTIONAL
  </policySet>
</policyCombinationStatement>
```

The <policyDesignator> element specifies a policy to include, using one of various ways of referring to a policy. There can be multiple <policyDesignator> elements in a <policyCombinationStatement>. The "combiner" specifies how the various policies are to be combined to produce a result.

Champion: Anne

Status: Closed

ISSUE:[PM-2-02: Target Specification]

According to the current schema each applicable policy can have multiple targets, each of which is an action and a URI identifying a set of resources (possibly with a transfer function to support wildcards). One may want to specify the target with reference to resource attributes (e.g., this policy applies to all files older that two years). How can I specify this?

[Tim] A different transform algorithm is all that is required. In the example, the "classification" is "older than two years", and the transform algorithm specifies how to deduce the age of a file.

Simon will present counter deductions to Anne 's proposal at the F2F

Potential Resolutions:

Ernesto suggests that this issue only mention retrieval of distributed policies and should be updated to reflect the recent discussion and Anne's proposal (See PM-1-01A) about policy combination. Anne volunteers to extend its wording in order to include policy combination as well.

Anne: [This note has to do with the syntax for expressing "applicability" of a single policy, and not with the logical rules for combining an inapplicable policy with other policies!!]

We currently allow a <target> element predicate in <applicablePolicy> element. The purpose of this element is to allow a PDP (or its agent, a PRP) to eliminate policies efficiently if they do not apply to the current authorizationDecisionQuery. Such an element can be used to index policies by Subject or Resource/Action (where some policies will need to be indexed under both Subject and Resource/Action, and some policies will apply to all Subjects and/or Resource/Actions). The idea is that the <target> element predicate is simple to compute, and allows the PDP (or PRP) to narrow down the field of potentially applicable policies efficiently. The PDP (or PRP) can then perform more complex evaluations on the smaller remaining set of policies.

Since the <target> element needs to be a simple predicate that is efficient to compute, it is not sufficiently expressive to rule out all cases where the <policy> may not apply. For example, if the policy applies only to employees who are over 55 years of age, then there is no syntax currently for expressing this in the <target> element.

POTENTIAL RESOLUTION:

We need two levels of applicability predicate: one used for fast narrowing down of the set of potentially applicable policies (and used for indexing), and the second for fully expressing the conditions under which this policy is applicable.

The first level applicability predicate is our current syntax: a regular expression match on a Resource/Action and Subject. It is very simple to compute, and MUST return TRUE for every authorizationDecisionQuery to which the corresponding policy applies. It MAY return TRUE for an authorizationDecisionQuery to which it does not apply. This predicate might be called "indexApplicability" or "basicApplicability" or something similar.

The second level applicability predicate is an optional new element in the <applicablePolicy>. It may use any comparison of attributes and values that could be used in the policy itself. This predicate might be called "fullApplicability" or something similar. This second level predicate is optional because for many policies, only the first level predicate may be required to fully capture the exact set of conditions under which the policy applies.

A policy evaluation returns "NOT-APPLICABLE" if either the first level applicability predicate OR the second level applicability predicate evaluates to FALSE. The second level predicate need be computed ONLY IF the first level predicate evaluates to TRUE.

The <policy> element may assume that the first and second level applicability predicates have been evaluated to TRUE. This may save some duplicate predicates.

Champion: Simon G.

Status: Open

ISSUE:[PM-2-03: Meaningful Actions]

There are pairings <resource,actions> which are not meaningful (e.g., execute a PDF file) [Simon G.]. Should we control resource/action bindings in the language or refer to an external authority?

Potential Resolutions:

[Tim] The administrative model in Figure 9 deals with this question, placing it out of scope for the schema. If we do need to tackle this, I suggest leaving it for a later version.

[Tim] The XACML syntax shall not address the question of which actions are valid for a particular resource classification. This matter shall be left for implementations to solve in a non-standard way. [PM-2-03]

Proposed Resolution:

The XACML syntax shall not address the question of which actions are valid for a particular resource classification.

Champion: Simon G.

Status: Closed

ISSUE:[PM-2-04: Indexing Policy]

Also related to target are indexing issues and how to retrieve, given a request, the applicable policy for it [Tim].

Potential Resolutions:

[Tim] Section 6.4 of version 0.8 of the language proposal is reserved for tackling this question in the LDAP case. Do we need to tackle other cases?

[Tim] The XACML specification shall provide normative, but non-mandatory to implement, text that profiles LDAP for distribution of XACML instances. [PM-2-04]

[Tim] The XACML specification shall provide normative, but non-mandatory to implement, text that profiles "the Web" for distribution of XACML instances. [PM-2-04]

Champion: Tim

Status: Open

ISSUE:[PM-2-05: Ensuring Completeness]

The applicable policy is defined as the ``complete" set of policies that apply to a resource. How do I ensure completeness (meaning no two targets should intersect?)

Potential Resolutions:

[Tim] This is a job for the PRP and should (I think) be out of the scope for our specification. The PRP has to be configured with the names and locations of the PAPs whose policies it recognizes.

[Tim] The XACML syntax shall not address the question of ensuring that "applicable policy" is complete. This matter shall be left for PRP implementations to solve in a non-standard way. [PM-2-05]

Potential Resolution:

1. If a Base Policy is included in the Access Request, then that Base Policy is the only one that will be applied to the Access Request. Otherwise,

2. If a PDP has a single Base Policy, then the PDP's Base Policy specifies the complete <applicablePolicy> that will be used by that PDP in evaluating an Access Request. This <applicablePolicy> may actually be a tree of <applicablePolicy> statements, where additional statements are logically incorporated by the use of <referencedPolicy> predicates.

In this case, there are no overlapping targets. If the PDP's Base Policy has an empty "target" element, then all Access Requests are evaluated against the <policy>. If the Base Policy has a non-empty "target" element, then any Access Request that does not match the "target" returns a result of "NOT-APPLICABLE" (=SAML INDETERMINATE). If the Access Request matches the "target", then the result of the Access Request is the result of evaluating the <policy>.

3. If a PDP has multiple Base Policies, then the PDP must specify and publish its algorithm for deciding which Base Policies to evaluate, in which order, and how target overlaps are resolved.

Vote:

2/21 It was agreed that this could be closed, but the **resolution has to be worded to be consistent with the new glossary**. This it was not voted closed.

3/7 Discussed and is not ready to be closed

Potential Resolution:

[This proposal depends on the proposed resolution to PM-3-03 and PM-3-03A: each PDP will have one base <policyCombinationStatement> or <policyStatement>]

A PDP must have a single base policy, which may be either a <policyStatement> or a <policyCombinationStatement>. The combiner algorithm in this base policy, together with the tree of associated <policySet> and <ruleSet> declarations, specifies the complete set of rules that the PDP will use in evaluating an access decision request.

Champion: Pierangela

Status: Open

ISSUE:[PM-2-06:Encapsulation of XACML policy (was Policy Security)]

Resolution 4: An XACML "applicable policy" will contain its own security features (e.g. signature), rather than relying on an encapsulating saml assertion.

Potential Resolutions:

[Anne] XACML will be specified in two separate layers.

1. The first layer is the <applicablePolicy> syntax, and will contain no security provisions such as authentication (signature), integrity protection, or encryption.

2. The second layer is a specification of how the first layer can be embedded in another mechanism for security protection. The XACML TC will define such a mechanism using an encapsulating SAML assertion. OASIS members are free to propose other mechanisms, such as encapsulating an <applicablePolicy> inside an X.509 Attribute Certificate.

Implementations may be compliant with the first layer only, with both the first layer and with the XACML TC-defined second layer, or with the first layer and another specified mechanism for the second layer. Implementations must state which level of compliance they support.

Champion: Tim

Status: Open

ISSUE:[PM-2-07: valueRef type]

Resolution 5: XACML valueRef elements shall be of type "saml:AttributeValueType".

Potential Resolutions:

???

Champion: Tim

Status: Open

ISSUE:[PM-2-08: Outcome of policies and their combination]

[Probably related to several other issues]

Proceedings on the discussion started at the F2F meeting, it is noted that outcome of policies is not only YES or NO but can have an alternative ``not applicable'' value, to this another possible value ``error'' seems to be needed. Anne also reports on her proposal (previously circulated via emal) about the use of ``if ... then.. `` rule for expressing policies. In her proposal the ``IF'' identifies the request to which a rule applies, if a request satisfies that then if the boolean expression in the THEN part is satisfied the response is ``allow'' otherwise it is ``deny''. If the IF part is not satisfied the response should be ``not applicable''. There is a discussion on what ``not applicable'' means. Hal points out the need for a default policy, to be applied if no target applies to the request. Tim points out that if the PEP sends a request to the PDP the PDP should return an error. Hal says that SAML would return a msg saying "indetermined status". Ernesto proposes defining an order on these values so that boolean operators can be applied as usual (and and or retain the usual behavior as long as the values on which they operate are organized in a lattice). The discussion proceeds on the different types on values and on what the intended combination should be. For instance, what should be the result between ``not applicable'' AND ``true''. The multivalue scheme that Ernesto is thinking of captures 4 values: false, true, lack of information, and not applicable. Ernesto and Polar say they will be thinking more about a possible lattice. Pierangela notes that there appears to be confusion in the policy combination since the current proposal does not distinguish between predicate evaluation and policy outcome. A predicate (i.e., one condition appearing in a rule) can either evaluate ``false'' ``true'' or ``notknown'' (in case the attribute is not provided). A policy can instead provide answes like ``allow'' ``deny'' or ``don't care''. The way we deal with ``notknown'' predicate evaluation and ``don't care'' policy decisions should not be the same. It might be possible to combine predicate evaluation and policy evaluation (as Anne notes policies can be nested, so a policy could appear where a predicate can) but we must be careful on how we combine them. Also ``don't care'' in policy decision means that we allow a policy to speak out in three different ways (and we should have a way to express that), this is independent from the ``not know" in the predicate evaluation.

Proposed Resolution:

[This resolution is related to the proposed resolutions to PM-1-01-A, PM-1-05, PM-1-07, PM-2-01, PM-3-03, PM-3-03A]

The combiner algorithm to be used by a given <policyStatement> or <policyCombinationStatement> is specified using a URI. The algorithm associated with the URI MAY be descriptive text.

XACML will specify a small set of mandatory-to-implement combiner algorithms. Users are free to define other algorithms, although not all XACML-compliant PDPs will be able to apply them.

The combiner algorithm specifies how the associated <ruleSet> or <policySet> is combined, and what the outcome will be.

Champion: Ernesto/Polar

Status: Closed

Group 3: Policy Composition

Assuming an Applicable Policy can refer to several Policy elements, we need to answer the following questions:

ISSUE:[PM-3-01: Combining Policy Elements]

How are the Policy Element combined? For instance, we could support Boolean expressions of policies. E.g., if there are three policies by independent issuers, I can say ``P1 AND (P2 OR P3)? This could fit well in the multiple issuers scenario Anne was envisioning. Should this be part of the core of the extension (external URI [Michiharu])?

Potential Resolutions:

[Tim] We could add "policy" to the "sequence" in "rule". Then we would have to give policies unique identifiers, not just string names. Perhaps, we should add "applicable policy", instead of "policy".

[Tim] An XACML "applicable policy" will not reference external "applicable policies". However, it may "incorporate" external "applicable policies". [PM-2-01] [PM-3-01] [PM-5-03]

[Tim] An XACML "applicable policy" shall be capable of referencing an external "applicable policy", providing explicit rules for combining such policies. [PM-2-01] [PM-3-01] [PM-5-03]

Proposed Resolution:

PolicyCombinationStatement allows policy writers to specify arbitrary algorithm to combine one or more PolicyStatement and/or one or more PolicyCombinationStatement. A policySetCombiner attribute in the PolicyCombinationStatement is used to identify the combination algorithm. PolicyMetaData MAY be used to combine policies.

Champion: Michiharu

Status: Closed

ISSUE:[PM-3-02: Specifying Policy Outcome]

How the policy outcome should be specified. Possibilities are 2-valued (access decision is ``grant"/"deny") or 3-valued (policy outcome is ``grant"/"deny"/nothing). Note the ``nothing" means that no rule applies, to be solved according to default. (Related work on composition…?)

How does the PEP interpret the answer I don’t know?

Potential Resolutions:

[Tim] Ultimately, the PEP has to know whether or not to grant access. So, someone has to decide, and (by definition) it is the PDP. So, the "don't care" response isn't helpful. However, saml should have an error code to indicate that the PDP is not the appropriate PDP to render a decision on a particular request.

[Tim] The XACML specification shall specify when a PDP should return saml:decision attributes with the values "permit" and "deny". If the PDP is unable to render a decision, then a saml status code shall be returned. No decision value shall be supplied in this case. [PM-3-02]

Champion: Simon

Status: Open

ISSUE:[PM-3-03: multiple Base Policies]

Can a PDP have more than one Base Policy?

Potential Resolutions:

Alternative 1:

A PDP MAY have multiple Base Policies, but such Base Policies SHOULD have non-overlapping <xacml:target> elements. The XACML specification does not specify the order in which multiple Base Policies are evaluated, or the result if two or more Base Policies have overlapping <xacml:target> elements.

A PDP that has multiple Base Policies MUST publish its algorithm for the order in which Base Policies are evaluated and the result where two or more Base Policies have overlapping

<xacml:target> elements.

Alternative 2:

Base Policies have restricted <target> elements that are easily compared for overlap. In this alternative, the case where base policies overlap is an ERROR. Note that the 0.8 syntax favors this alternative and allows Alternative 3.

Alternative 3:

There is only one Base Policy. Either it has no <target>, and applies to all Resources or it has a <target> element that specifies the set of resources which this PDP is prepared to handle and returns NOT-APPLICABLE if a resource does match that target.

Potential Resolution:

A given PDP uses a single <policyCombinationStatement> or <policyStatement> as the root of its evaluation. The <target> element of this base policy specifies the set of resources, subjects, and actions that this PDP is prepared to handle. This <target> element MAY be universal (allSubjects, allResources, allActions). A PDP returns NOT-APPLICABLE if a request does not match the <target> in its base policy.

[NOTE: Separate issue PM-5-13 of whether this can be overridden by input from the PEP].

Champion: Anne

Status: Open

ISSUE:[PM-3-03A: default PDP result]

If no Base Policy applies to a given Access Request (i.e. all Base Policy evaluations return NOT-APPLICABLE), does the PDP return NOT-APPLICABLE (=SAML INDETERMINATE) to the PEP, or is the PDP configured with a default result to return (e.g. TRUE or FALSE)?

Potential Resolution:

If no Base Policy applies to a given Access Request, then the PDP returns NOT-APPLICABLE (=SAML INDETERMINATE) to the PEP.

Potential Resolution:

A PDP must have a single base policy, which may be either a <policyStatement> or a <policyCombinationStatement>. This base policy will always return a result, whether it is "permit", "deny", "NOT-APPLICABLE", or "Indeterminate".

Champion: Anne

Status: Open

ISSUE:[PM-3-04: Pseudo Code for Combiner Algorithms]

Shall XACML mandatory-to-implement combiner algorithms be described using some sort of formal language or pseudo-code? If so, what syntax shall we use?

Anne, Ernesto, Carlisle, and Tim recommended that some sort of pseudo-code be used. Java was suggested. Ernesto offered to research various standard pseudo-codes and make a recommendation.

Champion: Ernesto.

Status: Open

Group 4: Syntax

ISSUE:[PM-4-01: Triplet Syntax (was Syntactic Sugar)]

The current schema assumes authorizations are specified as a pre-condition which is an expression made of predicates on SAML attributes (conditions on principal, resource and environment can be interspersed), let's call it Option ``pre-cond'' [Carlisle, Tim, Anne, ...]. In the last conference call it was agreed to leave as an open issue whether to group conditions about principal, resource, and environment in three different elements, let's call it Option ``triplet'' [Michiharu, Ernesto, Simon, ...]. The argument for Option ``pre-cond'' is that there are predicates that involve both principal and resource attributes (e.g., an authorization that states that users can read the files they own). The counter-objection to this is that you can naturally include all predicates on resources in the resource condition element (which can also refer to principal attributes). The argument for the triplet is that it makes authorization specifications conceptually clearer and closer to current approaches.

[Tim] In the 0.8 schema, valueRef has an attribute to indicate the entity to which it applies (principal, resource, etc.). It only has to be consulted if the attribute type identifier is ambiguous.

Potential Resolutions:

[Tim] The XACML syntax will differentiate between model entities (principal, resource, etc.) in its attribute elements, rather than in its rule elements. [PM-4-01]

Champion: Pierangela

Status: Open

ISSUE:[PM-4-02: Policy names as URIs]

Policy names are strings. Should we make then URIs?

Potential Resolutions:

Proposed Resolution:

Policy names should be URIs.

Vote:

2/21 Everybody agreed we should close this, because policy names are URIs in the current spec. Then we noticed that actually Policy Identifiers are URIs and Policy Names are strings. Everybody agreed this is the way it should be. Nobody could think of a reason to have an name and an id which were both URIs. **The Committee voted to close this issue with a resolution to leave the name and id as they are (string and URI respectively.)**

Champion: Tim

Status: Closed

ISSUE:[PM-4-03: Required type in policy]

The "rec:patient/patientName" element is a complex type. So, how should we indicate the required type in the policy?

[From PM-4-09] This only allows for simple types. Do we need to support values of complex type?

Potential Resolutions:

???

Champion: Tim

Status: Open

ISSUE:[PM-4-04:syntax extension]

Issue: should this element be an extension point to which other policy syntaxes can be added?

Potential Resolutions:

Propose Resolution:

Close this issue. It is incompletely specified: which element? Extension issues are in a separate section.

Vote:

The TC voted to close this issue as a matter of housekeeping and take up specific proposals for XACML extension points as separate issues.

Champion: Tim

Status: Closed

ISSUE:[PM-4-05:Policy Name a URI]

Issue: should we make policy name a URI?

Potential Resolutions:

See PM-4-02

Champion: Tim

Status: Closed as Duplicate

ISSUE:[PM-4-06:Comment element]

Issue: Should we include a "comment" element?

Potential Resolutions:

Proposed Resolution:

We should include a "comment" element.

Vote:

It was suggested that Annotation, which is built into XML schema be used instead. It was explained that this is for commenting Schemas, not instances. It was also pointed out that XML has a provision for imbedded comments. **The committee agreed to close this issue. The resolution is that an element called "Description" will be added to the schema and the text will say explicitly that the contents of this element MAY NOT affect policy evaluation in any way.**

Champion: Tim

Status: Closed

ISSUE:[PM-4-07:policy element in a rule]

Issue: Should we allow a policy element in a rule? Then the same schema could express the policy for combining policies. If so, should it be policy or applicable policy?

Potential Resolutions:

See PM-3-01

Champion: Tim

Status: Closed as Duplicate

ISSUE:[PM-4-08:XML elements include xsi:type]

Issue: Should we require XML elements compared in this way to include an xsi:type attribute?

Potential Resolutions:

???

Champion: Tim

Status: Open

ISSUE:[PM-4-09:complex types]

Issue: This only allows for simple types. Do we need to support values of complex type?

Potential Resolutions:

See PM-4-03

Champion: Tim

Status: Closed as Duplicate

ISSUE:[PM-4-10:preserve PAP identity]

Issue: Should the identities and/or signatures of the PAPs be preserved in the composed policy?

Potential Resolutions:

a <policyStatement> or <policyCombinationStatement> may be referenced as a saml assertion. In this case, the PAP identity, signature (if present), and other information is available to the associated combiner algorithm. Otherwise, the PAP identity is not preserved, and is not available to the associated combiner algorithm.

Champion: Tim

Status: Closed

Group 5: SAML Related

In the current schema attributes on resources and principals, which can be used in the Target (for resources) and in predicates, are retrieved using URIs pointing to SAML dataflow.

ISSUE:[PM-5-01: Non-SAML Input]

Can this mechanism be extended to point to non-SAML authorities as required in the Java environment [Sehkar]?

At a minimum, extending SAML expressions but broader to other authorities.

Potential Resolutions:

[Tim] The XACML specification shall be closely coupled to saml entities. However, the use of saml namespace identifiers is not intended to imply that all attributes must be retrieved from saml messages and assertions. [PM-5-01]

Champion: Sehkar

Status: Open

ISSUE:[PM-5-02: Wildcards on Resource Hierarchies]

How do we express wildcards on the resource hierarchies [Simon G.]?

The current schema includes ResourcetoClassificationTransform to this purpose. Is this sufficient?

Potential Resolutions:

[Tim] We should register an OASIS identifier for the use of regular expressions in this context.

[Tim] The XACML syntax shall use registered URIs to identify algorithms for processing resource classification wildcards. [PM-5-02]

Tied to outcome of resolution PM-5-14

Proposed Resolution:

Use "ResourceToClassificationTransform". Register a URI with OASIS for the use of regular expressions in this context. Other transform algorithms may be specified by the use of other URIs to be registered with OASIS.

Champion: Simon G.

Status: Ready to Close

ISSUE:[PM-5-03: Roles and Group Hierarchies]

Are roles and groups hierarchies available via SAML [Simon G.]? Hierarchies could be needed, in case of support of negative rules, for resolving conflicts based on more-specific-takes-precedence. Note: policy resolution conflicts fit well when the principal is a group, they may be difficult to apply in case of principal's expressions.

Potential Resolutions:

[Tim] An XACML "applicable policy" will not reference external "applicable policies". However, it may "incorporate" external "applicable policies". [PM-2-01] [PM-3-01] [PM-5-03]

[Tim] An XACML "applicable policy" shall be capable of referencing an external "applicable policy", providing explicit rules for combining such policies. [PM-2-01] [PM-3-01] [PM-5-03]

Proposed Resolution:

XACML will not support role and group hierarchies in the policy language. Attribute authorities may support role and group hierarchies.

Champion: Simon G.

Status: Closed

ISSUE:[PM-5-04: SAML Assertions URI]

From the schema it seems that expressions are predicates whose arguments are always URI or value. Are SAML assertions always URI?

Potential Resolutions:

[Tim] Attributes in saml assertions are identified by a namespace, which is a URI, and a name, which is a string.

Simon suggests that the current solution in general enough, as the URI+XPath combination specifies a schema (via the URI) and allows to retrieve a value (via the XPath). XPaths guarantee that values are uniquely identified. This technique smoothly applies not only to SAML but also to other formats like LDAP.

Hal observes that this is not always the case, as there may be attribute namespaces which are not URI.

Anne remarks that besides a pointer to the schema, a pointer to an instance is also needed. Simon agrees to provide a full explanation of this scenario at the F2F.

This issue conflates two separate issues:

1. Are SAML assertions always URI?
2. references to attributes in XACML predicates. (See new issue PM-1-04)

Proposed Resolution:

Attributes in SAML assertions are identified by a namespace, which is a URI, and a name, which is a string.

Champion: Simon

Status: Closed

ISSUE:[PM-5-05: XPath]

Use of Xpath for identifying SAML constructs and the use of Xpath operators

Potential Resolutions:

Simon clarifies that the position he will take is that while the use of Xpaths to extract nodeset is just fine, they do not make good values in expression. The solution in the current schema is cleaner.

Anne offers to look into the issue to provide an alternative point of view.

Champion: Simon

Status: Open

ISSUE:[PM-5-06: Multiple actions in single request]

In the SAML issues document, http://www.oasis-open.org/committees/security/docs/draft-sstc-core-discussion-01.doc

... Issue 5.1.15.2 seeks guidance on whether multiple "actions" can be specified in a single decision request.

Potential Resolutions:

[Tim] I feel that XACML should answer this question and send its conclusion in a liaison to SAML. My feeling is that the answer is "No". If "applicable policy" is to be identified with the resource/action pair, then multiple "applicable policies" are involved when multiple actions are involved. Much "cleaner" for there to be a single "applicable policy" for each decision request. And, therefore, a single action per decision request. It is no great hardship to submit multiple decision requests, in the event that you need a decision for each of several actions.

[Hal] Personally I am in favor of limiting this, but I will state the counter argument for the record. If the possible Actions correspond to what can be in the request, then this works fine. The only reason for multiple actions would be some sort of policy provisioning requirement. However, if the Actions are more like privileges or permission bits, and do not match allowable requests one for one, then some requests may require the AND or OR of several actions. I believe this is the motive behind suggesting multiple actions.

I don't see any rush on this as we are not close to proposing changes to the decision protocol yet.

Champion: Tim

Status: Open

ISSUE:[PM-5-07: Delegation]

[Polar] Has anybody thought about how delegation can be reasoned about in XACML? It appears that SAML only asserts a flat list of attributes with a single principal, or am I off base here? Can I support policies on such operations as:

Paul for Peter says debit Peter's account?

Which mean that Paul (or some other party trusted to do so) has issued Paul the authorization to act on behalf of Peter, in this case to access Peter's account. Or such things, like WebServer quoting JohnDoe says lookup in customer database. Where the WebServer may be trusted to authenticate JohnDoe, but no such proof is necessary other than the WebServer merely claiming to be acting on JohnDoe's behalf?

Potential Resolutions:

[Hal] With regards to SAML, the Access Decision Request was deliberately kept simple with the idea that XACML would give us the tools to do the job properly. I have proposed (see my use cases) that XACML not only be able to express policies, but the method of expressing policy inputs be rolled back into the SAML Access Decision Request (and Assertion).

In my opinion, XACML policies should be able to contain predicates about zero or more of the following subjects:

Requestor Subject

Recipient Subject (can be different from requestor)

Intermediary Subject (can be more than one for a given request)

I propose a single construct for Subjects and their attributes and some kind of modifier indicating the type (refrain from using "role" here) of subject.

[Tim] Delegation could be expressed in attribute assertions. The very issuance of an attribute assertion is a form of delegation. So, XACML should not have to concern itself with the process by which an entity obtained an attribute.

Champion: Polar/Hal

Status: Open

ISSUE:[PM-5-08: saml;Action is a “string”]

These are some of the potential SAML issues. Most of them were found when attempting to write J2SE policy files in XACML syntax. Further discussion is needed on these issues.

saml:Action is currently specified as a "string". Making Action an abstract type would allow it to be extended. This would allow the content model to be defined by a schema external to the SAML spec.

Thus what constitutes an action could be determined by the J2SE schema.

Potential Resolutions:

[Toshi] In SAML, saml:Action is used only in saml:Actions and saml:Actions have Namespace as an attribute. So it is possible to write action(s) such as:

```
<saml:Actions Namespace="urn:J2SEPermission:java.io.FilePermission">
    <saml:Action>write</saml:Action>
</saml:Actions>
```

or

```
<saml:Actions Namespace="urn:J2SEPermission">
   <saml:Action>java.io.FilePermission:write</saml:Action>
</saml:Actions>
```

But it will be useful if we can write something like:

```
<saml:Action>
    <J2SEPermission class="java.io.FilePermission">write</J2SEPermission>
</saml:Action>
```

Champion: Sekhar

Status: Open

ISSUE:[PM-5-09: saml;AuthorizationQuery requires actions]

If actions are optional for XACML, then why should <saml:Actions> be required in <saml:AuthorizationQuery> ? Both the wording in the SAML assertions draft as well as the SAML schema places such a requirement. saml:Actions should be optional in the AuthorizationQuery to accommodate queries without actions. At least for now, I don't anticipate this as an issue for J2SE.

Potential Resolutions:

[Toshi] In the latest SAML spec (core-25), AuthorizationDecisionQuery element has Resource attribute and Actions element and both of them are "required". Does this cause many problems?

(Resource attribute is "optional" for AuthorizationDecisionStatement element.)

As for J2SE case, I think there is an issue in terminology.

Champion: Sekhar

Status: Open

ISSUE:[PM-5-10: single subject in AuthorizationQuery]

[editor note: Is this issue covered somewhere else?]

saml:AuthorizationQuery currently only contains a single Subject. While a saml:Subject can support multiple NameIdentifier or SubjectConfirmation or AssertionSpecifier elements, it is required that they all belong to the same principal. So a single subject cannot be used for unrelated principals. In J2SE, there is a need to base access control on multiple principals which are not related and this therefore points to a need for more than one Subject in the saml:AuthorizationQuery

Potential Resolutions:

The way out of this appears to be extend SubjectQueryAbstractType.

Champion: Hal

Status: Open

ISSUE:[PM-5-11:XACML container in SAML]

Issue: should we use a SAML assertion as a container for an XACML applicable policy?

Potential Resolutions:

a SAML assertion MAY be used as a container for an XACML <policyStatement> or

<policyCombinationStatement>. The policy combiner MAY ignore the container elements, or MAY reference them in making its decision.

Champion: Tim

Status: Closed

ISSUE:[PM-5-12:derive attribute from saml:AttributeValueType]

Issue: Should we derive the attribute from saml:AttributeValueType? This seems to make sense, but the resulting attribute will have to become an element, with start and stop tags, making it larger and less readable.

Potential Resolutions:

???

Champion: Tim

Status: Open

ISSUE:[PM-5-13: Base Policy supplied as part of AuthorizationDecisionQuery]

Some PEPs have knowledge of the policy associated with a resource (example: a typical FileSystem knows the ACLs associated with a file or directory). To support this case, can a Base Policy or <referencedPolicy> be supplied as part of the SAML AuthorizationDecisionQuery?

Possible Resolutions:

Default policy:

A Base Policy or <referencedPolicy> for evaluating a particular Access Request may be specified as part of the Access Request. If a PDP has no Base Policy(s), then the result of evaluating an Access Request that does not specify a Base Policy to use is NOT-APPLICABLE (=SAML INDETERMINATE).

Champion: Anne

Status: Open

ISSUE:[PM-5-14: Resource Structure]

Simon proposes that the resource be written in a request-independent manner. The point that Simon makes in that while in SAML the resource is just a string, XACML should suggest a structure.

Hal comments that while it is good to retain a simplified structure, we should not be tied to

SAML as a specific way of expressing requests. In other words, we need to be compatible with SAML, but should not be tied to it. Carlisle, replies that we actually have that in the charter. Hal says we should be compliant, but we should ask SAML to define a more sophisticated request.

Simon says that the SAML way of expressing resources as a string is limited. For instance, what is the resource in case of XML documents? How do i go fine grained?

Ernesto comments that we should not have a sophisticated resource encoding if SAML does not support it. This can be a parallel effort to influence the next version of SAML.

Potential Resolutions:

Champion: Simon

Status: Open

ISSUE:[PM-5-15: Attribute reference tied to object]

Simon comments that attribute reference should be tied to the object. It's a question of tight coupling or loose coupling of the policy with the request. (This issue will be discussed in relationship with PM-5-14)

Potential Resolutions:

Champion: Simon

Status: Open

ISSUE:[PM-5-16: Arithmetic Operators]

The issue was discussed at the F2F where Sekhar said he would have looked at it. Sekhar reports that he could not complete it. Hal comments that we will need black box functions. for instance matching a subject requestor to something in a record that requires some sort of private functions: no set of simple operators that we can define that will be good enough. Ernesto, while agreeing on this, comments that it would be useful to have at least the simplest arithmetic operators be part of the language.

Potential Resolutions:

Champion: Ernesto, Simon, Tim

Status: Open

ISSUE:[PM-5-17: Boolean Expression of rules]

The current proposal in the document that a policy could be a boolean expression of rules. Pierangela points out that semantics of such a boolean expression seems to be not clear and while boolean expressions (or rather AND and OR) seems to be needed for combining policies they seems not to be for combining rules within an elementary policy.

Proposed Resolution:

The <condition> element in a <rule> can be a Boolean expression of predicates. <rule>s are combined in a <policyStatement> using a "combiner" algorithm, which specifies how the results of the <rule>s are combined. Likewise, <policyStatement>s and other <policyCombinationStatment>s are combined in a <policyCombinationStatement> using a "combiner" algorithm, which specifies how the results of the <policyStatement>s and <policyCombinationStatement>s are combined. Some combiner algorithms may be expressed using boolean expressions, but other combiner algorithms will use other logic. A combiner algorithm MAY be expressed using descriptive text rather than a formal language or pseudo-code.

Champion: Pierangela

Status: Closed

Group 6: Predicate Cononicalization

ISSUE:[PM-6-01: SAML Assertions URI]

Values used in predicates can refer to various standard formats (e.g, X.509 [Anne]) that could make the predicates evaluation difficult. For instance, if a principal's name is expressed in X.500 syntax you cannot compare it against a simple string. How do we make the representations canonical?

Potential Resolutions:

[Tim] Policy environments have to use consistent type definitions for the attributes they use.

Champion: Anne

Status: Open

Group 7: Extensibility

ISSUE:[PM-7-01: XACML extensions]

XACML Extension Model that defines what portion of the XACML specification is a core and to what extent the XACML specification can be extended. Based on this proposal, XACML policy administrators can represent much broader access control policies by extending the core portion of the XACML specification.

This extension model is designed to support an XACML extensibility property stated in the XACML charter. This proposal is based on the current language proposal document but includes several modifications.

Potential Resolutions:

See http://lists.oasis-open.org/archives/xacml/200112/msg00076.html

Champion: Michiharu

Status: Open

Group 8: Post Conditions

This group was created out of issues raised in Michiharu's proposal for post conditions. See Also Issues PM-1-02 and PM-1-03 for more on post conditions

ISSUE:[PM-8-01:] (4.1) Internal v.s. external post conditions

Proposed Resolution:

XACML does not support any distinction between internal post condition and external post condition. It depends on the configuration of PEP and/or PDP.

Champion: Michiharu

Status: Closed

ISSUE:[PM-8-02:] (4.2) Mandatory v.s. advisory post conditions

Proposed Resolution:

XACML does not support any distinction between mandatory obligation and advisory obligation. The meaning of the obligation is determined in each application.

Champion: Michiharu

Status: Closed

ISSUE:[PM-8-03:] (4.3) Inapplicable

Proposed Resolution:

The obligation is not returned to PEP when the authorization decision is determined as inapplicable or indeterminate.

Champion: Michiharu

Status: Closed

ISSUE:[PM-8-04:] (4.4) Base policy v.s. policy reference

The post conditions CAN be specified in the base policy as well as the policy reference. When the policy reference returns one or more post conditions, the base policy MUST deal with the returned post conditions. The possible processing rule is the following (this is subject to change):

> 4.4.1 Boolean expression handling
> In the base policy, the processor MUST determine whether the condition holds or not regardless of the post condition.
>
> 4.4.2 Post condition handling
> If the condition holds, the processor gathers all the post conditions that are attached to the TRUE conditions. If the condition does not hold, the processor gathers all the post conditions that are attched to the FALSE conditions.
>
> 4.4.3 Return final decision
> After gathering all the post conditions, the processor returns Grant or Deny permission with corresponding post condition(s).

Proposed Resolution:

The obligation is specified in both policyStatement and policyCombinationStatement. The scope of the obligation is defined in ISSUE: PM-1-02 as "The set of obligations returned by each level of evaluation includes only those obligations associated with the effect element being returned by the given level of evaluation. For example, a policy set may include some policies that return Permit and other policies that return Deny for a given request evaluation. If the policy combiner returns a result of Permit, then only those obligations associated with the policies that returned Permit are returned to the next higher level of evaluation. If the PDP's evaluation is viewed as a tree of policyCombinationStatements, policyStatements, and rules, each of which returns "Permit" or "Deny", then the set of obligations returned by the PDP will include only the obligations associated paths where the effect at each level of evaluation is the same as the effect being returned by the PDP."

Champion: Michiharu

Status: Closed

ISSUE:[PM-8-05:] (4.5) How to return post conditions via SAML

Post conditions are stored in <condition> element of SAML authorization decision assertion. XACML provides a namespace for storing post conditions. (It would be an unbounded sequence of <operation> element.)

Toshi: Though using <Conditions> element might be one option, I think it is preferable to place post conditions in <Statement> (<AuthorizationDecisionStatement>) element (but there is no room for it now).

Michiharu: First I had the same idea and if such modification is accepted by SAML, that would be the ideal way to take. Actually, I tried to find alternative solution that might work under a certain assumption. AuthorizationDecisionStatement may include validity period such as "from 1 March to 31 March" in <Conditions> element in some cases. But access decisions returned by XACMLed PDP will not generate such restriction from the discussion in XACML so far. Thus, I thought that <Conditions> element can be used for post-conditions. From the PEP viewpoint, it is easy to distinguish AuthorizationDecisionStatement generated by XACMLed PDP from one generated by other component by looking <Issuer> element etc. But I am not confident with this usage.

Bill: In my mind, this puts the responsibility of appropriate *action* on the PEP; the PDP is only concerned with *decisions*, and those decisions are finite (within the scope of the decision making process). personally, i think that we should proceed with the assumption that SAML will be open to modifications to their specification--if our reasoning is sound i do not see why we would not be able to garner support for adoption.

Toshi: When we put post-conditions in <Conditions> element, we must extend SAML <Condition> element (I noticed it today). Then how about extending SAML <AuthorizationDecisionStatement> element? SAML allows to extend it. It will look like as follows:

```
<element name="AuthorizationDecisionWithPostConditionStatement"
    type="xacml:AuthorizationDecisionWithPostConditionStatementType"/>
<complexType name="AuthorizationDecisionWithPostConditionStatementType">
  <complexContent>
    <extension base="saml:AuthorizationDecisionStatementType">
      <sequence>
        <element ref="xacml:PostConditions"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Bill: the difference between these approaches appears to be where the PDP's responsibility ends. as i see it, if you use the <Condition> element approach, the PDP still maintains some level of implied responsibility for seeing that this condition is met ('registering in the post-condition conponenet'). on the other hand, extending the <AuthorizationDecisionStatement> element releases this responsibility to the PEP ('i issue a GRANT, however i base that upon the stipulation that *you, the PEP*, will discard this access 30 days hence.')

either way, the GRANT is issued without waiting 30 days, but the latter approach appears more in line with the concept of this being a 'stipulation' or 'constraint' rather than a 'condition' (which to me implies that it's completion is requried to generate the GRANT -- clearly not the case here)

obviously, a level of implied trust is inherent in this approach (hey, if you can't trust the PEP who can you trust? :o); this is not enforceable by the PDP, however if the behavior of the PEP is to DENY unless it can interpret (and fulfill) the stipulation, it sees that you would have a workable solution.

Anne: think I agree with Bill's position on this: the PDP should be just an evaluation engine. It can not be held responsible for enforcing any actions as a result of the evaluation. Post conditions, if we use them, should just be values that are returned to the PEP and are meaningful only to the PEP. It is up to the PEP to enforce them.

I think the semantics of post conditions are hard to manage in access control unless we want the PDP to be far more than an evaluation engine.

The one strong argument for PDP-enforced post conditions I have heard is that certain actions should be logged by the PDP, showing exactly how the result was obtained. I think this can probably be an implementation feature for a PDP, managed by PDP configuration and outside of the scope of XACML. It is not part of a policy.

Post conditions are stored in <condition> element of SAML authorization decision assertion. XACML provides a namespace for storing post conditions. (It would be an unbounded sequence of <operation> element.)

a <saml:Condition> element is a child element of a <saml:Assertion> element, not a <saml:AuthorizationDecisionStatement>. If we allow multiple decisions per assertion, then <saml:Condition> is not a suitable place for our <xacml:obligations> element.

Proposed Resolution:

Here is an authorization decision syntax that returns obligation(s). SAML AuthorizationDecisionStatement is extended to include xacml:obligations element by type extension. "samle" namespace prefix is used to indicate SAML extension for the decision assertion with obligation. Note that the following example just shows the overview for simplicity.

```
<saml:Assertion>
  <saml:AuthorizationDecisionStatement Resource="aaa" Decision="Permit"
xsi:type="samle:AuthorizationDecisionStatementWithObligations">
  <saml:Subject>
   <saml:NameIdentifier SecurityDomain="aaa" Name="Alice"/>
  </saml:Subject>
  <saml:Actions Namespace="http://www.oasis-open.org/xmlactions">
   <saml:Action>Read</saml:Action>
  </saml:Actions>
  <xacml:obligations>
   <xacml:obligation obligationId="myId">
    ...
   </xacml:obligation>
  </xacml:obligations>
  </saml:AuthorizationDecisionStatement>
</saml:Assertion>
```

The following "saml" schema fragment defines an authorization decision with obligations.

```
<complexType name="AuthorizationDecisionStatementWithObligations">
 <complexContent>
  <extension base="saml:AuthorizationDecisionStatementType">
   <sequence>
    <element ref="xacml:obligations"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

Champion: Michiharu

Status: Ready To Close

ISSUE:[PM-8-06:] (4.6) When to execute post condition

While post condition implies that specified operations must be dealt with prior to the requested access, it does not necessarily mean that the specified operations must be executed synchronously. Taking the obligatory operation usage scenario in 1.2 for example, it is impossible to execute "delete-in-90days" post condition prior to the requested access. It would be reasonable if such operation is queued in the application and guaranteed to be executed later.

Proposed Resolution:

When and how PEP executes obligation depends on each application. XACML (as PDP) does not assume any specific semantics. While obligation implies that specified operation must be dealt with prior to the requested access, it does not necessarily mean that the specified operations must be executed synchronously. Taking the obligatory operation usage scenario like "customers can register themselves with their private information provided that such information is deleted in 90 days--- obligation is delete-in-90days", it is impossible to execute "delete-in-90days" obligation prior to the requested access. It would be reasonable if such operation is queued in the application and guaranteed to be executed later.

Champion: Michiharu

Status: Closed

ISSUE:[PM-8-07:] (4.7) Extension point

Proposed Resolution:

XACML SHOULD support extension point in the post condition specification and semantics. It includes the process of how to determine the post condition. One example is that the processor selects the post condition that is attached to the rule of the highest priority.

Extension point of obligation is 1. obligationId in policyStatement or policyCombinationStatement and 2. ruleSet combiner or policySet combiner. This allows policy writers to specify arbitrary identifier of the user-defined obligation and to specify the semantics of how obligation is computed in response to the access request.

Champion: Michiharu

Status: Closed

Miscellaneous Issues

Group 1: Glossary

ISSUE:[MI-1-01: Consistency]

Pierangela mentioned something discussed in PM group that may not coincide with glossary concerning pre and post conditions.

Proposed Resolution:

Any glossary concerns should be resolved as part of the resolution for the particular issue in the PM group.

Champion: Pierangela

Status: Closed

ISSUE:[MI-1-02: Definition of Policy vs. Rule]

In our glossary, "rule" is a predicate or a logical combination of predicates, and "policy" is a set of rules (which I've always taken to be a logical combination of rules, although the glossary doesn't explicitly say so and, from what Pierangela was saying yesterday, she took it to be a simple "OR" of rules).

In the proposal that I posted last Friday, I tried to make a couple of other distinctions: a rule does not have an applicability or target element, whereas a policy does; and a rule has an explicit grant/deny indicator, whereas a policy does not.

But in yesterday's call, Simon said that in his mind a rule does have an applicability element (a R-A-S triple, which may be a simplified version of the predicates contained in the rule). Furthermore, he thinks that a policy should have a grant/deny indicator (or at least grant, for now). And, as I mentioned above, Pierangela questioned whether there is any need for a policy to have a combination of rules (i.e., either it is just a combination of predicates, or it is implicitly understood that they are combined in an OR). Finally, Simon suggested that the smallest individual unit specified by XACML should be a policy.

So now I really don't understand the difference between "policy" and "rule". How are they different? Do we need to distinguish between them? Do we need separate syntax for them? Why not forget about rules altogether and say that, for XACML, a logical combination of predicates, with a (possibly simplified) applicability or target element, and with an explicit grant/deny indicator, *is* a policy. No mention of rules whatsoever (except possibly in the "Related Terms" section that follows the glossary).

Is this acceptable, or is there an important distinction that needs to be maintained in the syntax?

Note 1) I think we still need to retain the concept of a higher-level policy (e.g., a base policy) that specifies a logical combination of sub-policy results. The sub-policies may be included or referenced.

Note 2) I think it would be useful to include the concept of a meta-policy that specifies a logical combination of predicates about policy (e.g., grant/deny, or issuer, or issue date, or whatever). I don't know how else to be able to say general things like "policies from this authority always override policies from that authority", or "denies always override grants", or "policies issued in the past month always override older policies".

Proposed Resolution:

A "rule" is the smallest unit from which a "policy" is composed. A "rule" uses predicates that refer to attributes and values.

A "policy" is a combination of rules or other policies. A combination of rules is called a <policyStatement>. A combination of <policyStatement>s or other <policyCombinationStatement>s is called a <policyCombinationStatement>. A policy is the smallest administrative unit in XACML, and is the smallest unit that can be signed. A policy does not refer to attributes and values, but only to combinations of rules or other policies.

Champion: Carlisle

Status: Closed

ISSUE:[MI-1-03: Definition and purpose of Target]

There seems to be some confusion, at least in the mind of the scribe ;-) but it seems to be shared by others, on the concept and the use of target. Carlisle points out that the target essentially represent a ``condition" on the access requests to which the attached policy refers and those it provides a way to avoid going into the evaluation of policies that do not apply to the request. Intuitively, a target is like a condition that should have appeared in AND with the others in all the rules in the attached policy. Hal says that target can be useful in many real life situations for specifying policies as the administrator explicitly stated to what set of access a set of rules applies.

Proposed Resolution:

a <target> element consists of three predicates over elements in a SAML access decision request: one over Subject, one over Resource, and one over Action. Any of these predicates may be universal in that they may result in "true" for "anySubject", "anyResource", or "anyAction".

Tthe <target> element in a <rule>, <policyStatement>, or <policyCombinationStatement> has two purposes. First, it allows <rule>s, <policyStatement>s, and policyCombinationStatement>s to be indexed based on their applicable subject, resource, and/or action. Second, it allows a PDP to quickly and efficiently reduce the set of <rule>s, <policyStatement>s, and <policyCombinationStatement>s that must be evaluated in response to a given access decision request.

These intended purposes place three restrictions on what can be included in a <target>. First, the predicates in a <target> must be very efficient to evaluate. Second, each predicate in a <target> must refer to only one of <subject>, <resource>, and <action> (for indexing purposes). Third, each predicate in a <target> must refer only to attributes that will always be present in a SAML access decision request, since a <target> must not return a result of "indeterminate".

In a <rule>, the <target> element is logically part of the <condition> element. Were indexing and efficiency not a concern, the tests in the <target> could be incorporated into the <condition>. The <target> element serves as the "first pass" test for whether the rule applies:

```
if (<target> == true) {
    if (<condition> == true) {
        return <effect>;
    }
}
return <not applicable>;
```

Champion: Anne

Status: Ready To Close

Group 2: Conformance

ISSUE:[MI-2-01: Successfully Using]

XACML definition of OASIS requirement to successfully use the specification

Potential Resolutions:

"Successfully Using the XACML Specification"

XACML is an XML schema for representing authorization and entitlement policies. However, it is important to note that a compliant Policy Decision Point (PDP) may choose an entirely different representation for its internal evaluation and decision-making processes. That is, it is entirely permissible for XACML to be regarded simply as a policy interchange format, with any given implementation translating the XACML policy to its own local/native/proprietary/alternate policy language sometime prior to evaluation.

A set of test cases (each test case consisting of a specific XACML policy instance, along with all relevant inputs to the policy decision and the corresponding PDP output decision) will be devised and included on the XACML Web site.

In order to be "successfully using the XACML specification", an implementation MUST, for each test case, have a "policy evaluation component" that can consume the policy instance and the inputs and produce the specified output.

Furthermore, the implementation MUST have a "policy creation component" that allows it to generate schema-valid XACML policy instances that can be consumed/processed by other PDPs.

Note that, aside from the XACML policy instance itself, all PDP inputs and outputs MUST be SAML-compliant (i.e., conform with the assertions and protocol messages defined in the SS-TC SAML specification), although other syntaxes/formats for the PDP input and output MAY be supported in addition to this.

Champion: Carlisle

Status: Closed

Group 3: Patents, IP

ISSUE:[MI-3-01: XrML]

[Ernesto] As I recollect, OASIS requested us to evaluate whether any XACML specification might fall in the scope of patents held by others. I quote from a Dec 13th addition to announcements regarding Xerox's XrML:

(http://xml.coverpages.org/xrml.html) :

"ContentGuard's strategy appears to be to make money by licensing the technology -- whatever some outside body defines it to be. It can do this because its patents cover the idea of a rights language in general, no matter what the specifics of the language are".

I know XrML has already been mentioned in our discussions from the technical point of view, but the wording of this announcements makes me suspect that we should explore the matter further from the patents' point of view.

Potential Resolutions:

Oasis has a specific IPR policy and ContentGuard needs to make Oasis aware of any IP as it relates to XACML or other technical committees in accordance with that policy.

[Hal] Paragraph (C) of OASIS.IPR.3.2. makes the following points:

If OASIS knows about something they "shall attempt to obtain from the claimant of such rights a written assurance ..."

However, "results of this procedure shall not affect advancement of a specification..."

Except that "The results will, however, be recorded..." and "...may also direct that a summary of the results be included in any OASIS document published containing the specification." It also says elsewhere that they will not go out of their way to find IPR that has not been drawn to their attention.

Champion: Ernesto

Status: Open

Group 4: Other Standards

ISSUE:[MI-4-01: RuleML]

Should XACML look at RuleML?

[Edwin] XACML folks, Since XACML is about defining "rules" for Authorization -- would it make sense to leverage work done by the RuleML folks?

RuleML folks, You may want to checkout XACML as an application of RuleML. Here is a standard that will be real within the next year!]

Potential Resolutions:

The issue is a generic suggestion about XACML to be a possible application of a general setting for rule representation, RuleML.

Anne proposes that at the F2F every suggestion of taking into account related languages should be mandatory accompanied by a presentation

After a brief discussion on RuleML, the issue is voted closed. It should be deleted from the next version of the issues document

Champion: Edwin

Status: Closed

ISSUE:[MI-4-02: RAD]

Should XACML look at RAD?

[Polar] In response to some query about the expressiveness of evaluation of policies from different places, I would like to point the group to the CORBA Resource Access Decision specification (RAD).

http://www.omg.org/cgi-bin/doc?formal/01-04-11.pdf

and we may want to include it the document repository. It has in it an Access Decision model in which not only policies are located, but also, a policy evaluation combinator is located for a particular resource. Note, there is no language component to this specification.

However, it does present a model by which policy can be distributed and evaluated. A combinator, which has an interface operation of "evaluate_policies" takes the list of located policies for the resource, the attribute list of the subject, and the operation (i.e. Action) on the resource) and evaluates the decision.

That way, depending the semantics of the combinator you choose for the resource, your combinator may choose to ignore, or evaluate only some policies based on the evaluations of other policies.

Potential Resolutions:

Polar will bring that one to the discussion, with special reference to policy combination.

Champion: Polar

Status: Open

ISSUE:[MI-4-03: DSML]

Transformations from XACML to DSML

[Gil] Since the last time we talked I had the chance to play with DSML a little. It seems to me that it is theoretically possible to transform an XACML policy document into a DSML document and import that document into LDAP. The DSML document could contain elements that described the (LDAP) schema necessary to store the authorization policy entries in case the target LDAP didn't already have this schema. It is also possible to export some LDAP entries into a DSML document and transform that DSML document in XACML.

What I don't know (having nothing more than a cursory understanding of XSL/XSLT) is how difficult such transformations would be and if there are any "gotchas" that would keep this from really working.

Potential Resolutions:

[Gil] What I think the XACML spec should do is:

1.) Describe the LDAP schema necessary to store authorization policies. This should be done in "LDAP fashion" with dn's, classnames, etc.

2.) (if possible) Provide the XSLT necessary to transform XACML to DSML and vice versa.

That way people who don't want to be bothered with DSML can work out their own way to store and retrieve XACML data to and from the defined schema.

Champion: Gil

Status: Open

ISSUE:[MI-4-04: Java Security Model]

Hal says he is not clear about whether XACML should be able to represent the Java security model. Gil comments that XACML would be limited if it cannot express it. Hal notes that what XACML should be able to represent are the same requirements that Java security model represents, but not necessarily in the same way (i.e., representing the same authorizations).

Potential Resolutions:

???

Champion: Sekhar

Status: Open

Document History

- 7 Jan 2002 First Version Published
- 21 Jan 2002 Major edits and additions. Every open item updated.
- 18 Feb 2002 Edits based on F2F and Anne’s edits
- 27 Feb 2002 Edits based on 2/21 voting and post condition issues
- 8 Mar 2002 Version 5 released but title page had version 4 information
- 27 Mar 2002 Closed issues updated from F2F and Policy Model Calls