



XACML

XACML - Topics

- Goals
- Approach
- Examples
- Summary

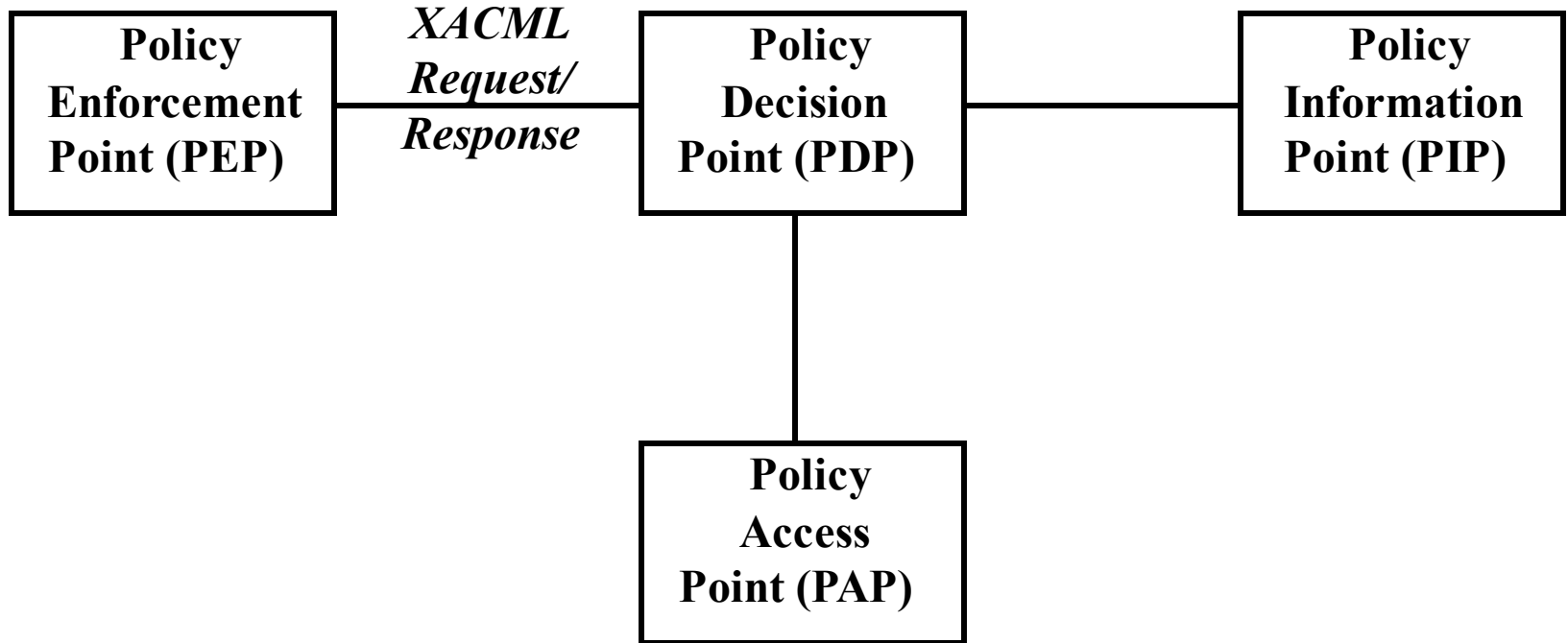
Goals

- Define a core XML schema for representing authorization and entitlement policies
- Target - any object - referenced using XML
- Fine access control grained control
- Access control based on subject and object attributes
- Access control based on the object contents; if the object is not an XML document, the object attributes can be used
- Consistent with and building upon SAML

XACML – Key Aspects

- General-purpose authorization policy model and XML-based specification language
- XACML is independent of SAML specification
- Triple-based policy syntax: <Object, Subject, Action>
- Negative authorization is supported
- Input/output to the XACML policy processor is clearly defined as XACML context data structure
- Input data is referred by XACML-specific attribute designator as well as XPath expression
- Extension points: function, identifier, data type, rule-combining algorithm, policy-combining algorithm, etc.
- A policy consists of multiple rules
- A set of policies is combined by a higher level policy (PolicySet element)

XACML Protocol



XACML Protocol

- When a client makes a resource request upon a server, the PEP is charged with AC
- In order to enforce AC policies, the PEP will formalize the attributes describing the requester at the PIP and delegate the authorization decision to the PDP
- Applicable policies are located in a policy store, managed by the PAP, and evaluated at the PDP, which then returns the authorization decision
- Using this information, the PEP can deliver the appropriate response to the client

XACML Protocol

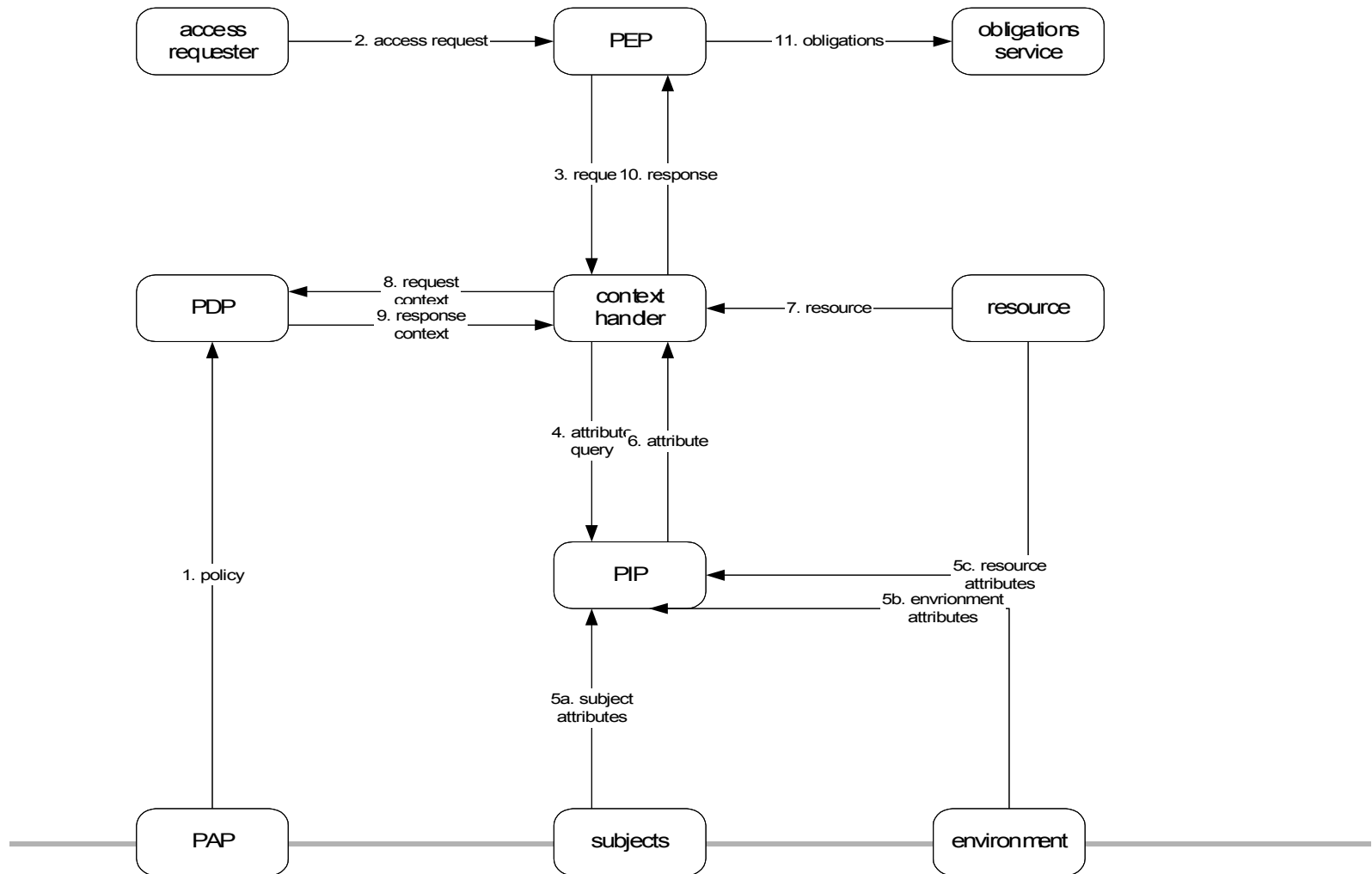
1. The *Policy Administration Point* (PAP) creates security policies and stores these policies in the appropriate repository.
2. The *Policy Enforcement Point* (PEP) performs access control by making decision requests and enforcing authorization decisions.
3. The *Policy Information Point* (PIP) serves as the source of attribute values, or the data required for policy evaluation.
4. The *Policy Decision Point* (PDP) evaluates the applicable policy and renders an authorization decision.

Note: The PEP and PDP might both be contained within the same application, or might be distributed across different servers

XACML Protocol

- XACML Request
 - Subject
 - Object
 - Action
- XACML Response
 - Permit
 - Permit with Obligations
 - Deny
 - NotApplicable (the PDP cannot locate a policy whose target matches the required resource)
 - Indeterminate (an error occurred or some required value was missing)

Data Flow Model



Data Flow Model

1. PAPs write **policies** and **policy sets** and make them available to the PDP. These **policies** or **policy sets** represent the complete policy for a specified **target**
2. The access requester sends a request for access to the PEP
3. The PEP sends the request for **access** to the context handler in its native request format, optionally including **attributes** of the **subjects**, **resource**, **action** and **environment**
4. The **context handler** constructs an XACML request **context** and send it to the PDP
5. The PDP requests any additional **subject**, **resource**, **action**, and **environment attributes** from the context handler
6. The context handler requests the attributes from a PIP
7. The PIP obtains the requested **attributes**
8. The PIP returns the requested **attributes** to the context handler
9. Optionally, the context handler includes the **resource** in the context

Data Flow Model

10. The context handler sends the requested **attributes** and (optionally) the resource to the PDP. The PDP evaluates the **policy**
11. The PDP returns the response **context** (including the **authorization decision**) to the context handler
12. The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP
13. The PEP fulfills the **obligations**
14. (Not shown) If **access** is permitted, then the PEP permits **access** to the **resource**; otherwise, it denies **access**

XACML Schemas

Request Schema

Request

Subject

Resource

Action

Policy Schema

PolicySet (Combining Alg)

Policy* (Combining Alg)

Rule* (Effect)

Target

Subject*

Resource*

Action*

Environment

Effect

Condition

Response Schema

Response

Decision

Obligation*

Obligation*

Pag. 12

XACML Schemas

Request Schema

Request

Subject

Resource

Action

Policy Schema

PolicySet (Combining Alg)

Policy* (Combining Alg)

Rule* (Effect)

Subject*

Resource*

Action

Condition*

Obligation*

Response Schema

Response

Decision

Obligation*

Policies and PolicySet

- The key top-level element is the <PolicySet> which aggregates other <PolicySet> elements or <Policy> elements
- The <Policy> element is composed principally of <Target>, <RuleSet> and <Obligation> elements and is evaluated at the PDP to yield an access decision.
- Since multiple policies may be found applicable to an access decision, (and since a single policy can contain multiple Rules) Combining Algorithms are used to reconcile multiple outcomes into a single decision
- The <Target> element is used to associate a requested resource with an applicable Policy. It contains conditions that the requesting Subject, Resource, or Action must meet for a Policy Set, Policy, or Rule to be applicable to the resource.
- The Target includes a build-in scheme for efficient indexing/lookup of Policies.
- Rules provide the conditions which test the relevant attributes within a Policy. Any number of Rule elements may be used each of which generates a true or false outcome. Combining these outcomes yields a single decision for the Policy, which may be "Permit", "Deny", "Indeterminate", or a "NotApplicable" decision.

Policies and Policy Sets

- Policy
 - Smallest element PDP can evaluate
 - Contains: Description, Defaults, Target, Rules, Obligations, Rule Combining Algorithm
- Policy Set
 - Allows Policies and Policy Sets to be combined
 - Use not required
 - Contains: Description, Defaults, Target, Policies, Policy Sets, Policy References, Policy Set References, Obligations, Policy Combining Algorithm
- Combining Algorithms: Deny-overrides, Permit-overrides, First-applicable, Only-one-applicable

Overview of the Policy Element

```
<Policy>
  <Target>
    <Resources>
    <Subjects>
    <Actions>
    <RuleSet ruleCombiningAlgId = "DenyOverrides">
      <Rule ruleId="R1">
      <Rule ruleId="R2">
      ...
    <Obligations>
    <RuleSet>
  </Policy>
```

```
<Rule RuleId="R2"
      Effect="Deny">
  <Target>
```

```
<Rule RuleId="R1"
      Effect="Permit">
  <Target>
    <Resources>
    <Subjects>
    <Actions>
    <Condition>
  </Rule>
```


Combining Algorithms

- Policy & Rule Combining algorithms

Permit Overrides:

If a single rule permits a request, irrespective of the other rules, the result of the PDP is Permit

Deny Overrides:

If a single rule denies a request, irrespective of the other rules, the result of the PDP is deny.

First Applicable:

The first applicable rule that satisfies the request is the result of the PDP

Only-one-applicable:

If there are two rules with different effects for the same request, the result is indeterminate

Rules

- Smallest unit of administration, cannot be evaluated alone
- Elements
 - Description – documentation
 - Target – select applicable rules
 - Condition – boolean decision function
 - Effect – either “Permit” or “Deny”
- Results
 - If condition is true, return Effect value
 - If not, return NotApplicable
 - If error or missing data return Indeterminate
 - Plus status code

Target

- Designed to efficiently find the policies that apply to a request
- Makes it feasible to have very complex Conditions
- Attributes of Subjects, Resources and Actions
- Matches against value, using match function
 - Regular expression
 - RFC822 (email) name
 - X.500 name
 - User defined
- Attributes specified by Id or XPath expression
- Normally use Subject or Resource, not both

Rule Element

- The main components of the <rule> element are:
 - a <target>
 - the <target> element consists of
 - a set of <resource> elements
 - a set of <action> elements
 - an environment
 - the <target> element may be absent from a <rule>. In this case the <target> of the rule is the same as that of the parent <policy> element
 - an <effect>
 - Two values are allowed: “Permit” and “Deny”
 - a <condition>

Policy Element

- The main components of a <policy> element are:
 - a <target> element
 - the <target> element consists of
 - a set of <resource> elements
 - a set of <action> elements
 - an environment
 - the <target> element may be declared explicitly or may be calculated; two possible approaches:
 - Make the union of all the target elements in the inner rules
 - Make the intersection of all the target elements in the inner rules
 - a rule-combining algorithm-identifier
 - a set of <rule> elements
 - obligations

PolicySet Element

- The main components of a `<policyset>` element are:
 - a `<target>`
 - a policy-combining algorithm-identifier
 - a set of `<policy>` elements
 - obligations

A Policy Example

- The Policy applies to requests for the server called “SampleServer”
- The Policy has a Rule with a Target that requires an action of "login" and a Condition that applies only if the Subject is trying to log in between 9am and 5pm.
- Note that this example can be extended to include other Rules for different actions.
- If the first Rule does not apply, then a default Rule is used that always returns Deny (Rules are evaluated in order).

A Policy Example

```
<Policy PolicyId="SamplePolicy" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides">
  <!-- This Policy only applies to requests on the SampleServer -->
  <Target>
    <Subjects> <AnySubject/> </Subjects>
    <Resources>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">SampleServer
          </AttributeValue>
        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/> </ResourceMatch>
      </Resources>
    <Actions> <AnyAction/> </Actions>
  </Target>
```


A Policy Example

```
<!-- Rule to see if we should allow the Subject to login -->
<Rule RuleId="LoginRule" Effect="Permit">
<!-- Only use this Rule if the action is login -->
  <Target>
    <Subjects> <AnySubject/> </Subjects>
    <Resources> <AnyResource/> </Resources>
    <Actions>
      <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">login</AttributeValue>
        <ActionAttributeDesignator DataType=http://www.w3.org/2001/XMLSchema#string
          AttributeId="ServerAction"/>
      </ActionMatch>
    </Actions>
  </Target>
```

A Policy Example

```
<!-- Only allow logins from 9am to 5pm -->
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal"
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
      <EnvironmentAttributeSelector
        DataType="http://www.w3.org/2001/XMLSchema#time"
        AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/> </Apply>
    <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue> </Apply>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal"
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
      <EnvironmentAttributeSelector
        DataType="http://www.w3.org/2001/XMLSchema#time"
        AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
    </Apply>
    <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</AttributeValue> </Apply>
</Condition>
</Rule>
</Policy>
```

Condition

- Boolean function to decide if Effect applies
- Inputs come from Request Context
- Values can be primitive, complex or bags
- Can be specified by id or XPath expression
- Fourteen primitive types
- Rich array of typed functions defined
- Functions for dealing with bags
- Order of evaluation unspecified
- Allowed to quit when result is known
- Side effects not permitted

Functions

- Equality predicates
- Arithmetic functions
- String conversion functions
- Numeric type conversion functions
- Logical functions
- Arithmetic comparison functions
- Date and time arithmetic functions
- Non-numeric comparison functions
- Bag functions
- Set functions
- Higher-order bag functions
- Special match functions
- XPath-based functions
- Extension functions and primitive types

Request and Response Context

- Request Context
 - Attributes of:
 - Subjects – requester, intermediary, recipient, etc.
 - Resource – name, can be hierarchical
 - Resource Content – specific to resource type, e.g. XML document
 - Action – e.g. Read
 - Environment – other, e.g. time of request
- Response Context
 - Resource ID
 - Decision
 - Status (error values)
 - Obligations

XACML History

- First Meeting – 21 May 2001
- Requirements from: Healthcare, DRM, Registry, Financial, Online Web, XML Docs, Fed Gov, Workflow, Java, Policy Analysis, WebDAV
- XACML 1.0 - OASIS Standard – 6 February 2003
- XACML 1.1 – Committee Specification – 7 August 2003
- XACML 2.0 – Complete summer 2004
- XACML 3.0 – In progress