OASIS eXtensible Access Control Markup Language (XACML)

XML Community of Practice, 21 June 2006

Anne Anderson
Senior Staff Engineer
Sun Microsystems Laboratories

Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

Motivation for XACML

- Data more accessible today
- Regulatory requirements
- Protection at the enforcement level
- Protection of XML documents
- Standard language
  - Interoperability
  - Common policies, tools, analysis
  - Auditability

Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

XACML Overview

- XML-based
- Schemas for
  - Policies
  - Access decision requests
  - Access decisions
- Rich set of functions and datatypes for expressing policies
- Specification: how to evaluate policies against access decision requests

XACML Overview: standardization

- OASIS Standard
- XACML 1.0 approved Feb. 2003
- XACML 2.0 approved Feb. 2005, including Profiles:
  - Privacy
  - Role Based Access Control
  - OASIS Security Assertions Markup Language (SAML)
  - ...
- XACML 3.0 in progress
  - Policy-controlled policy administration

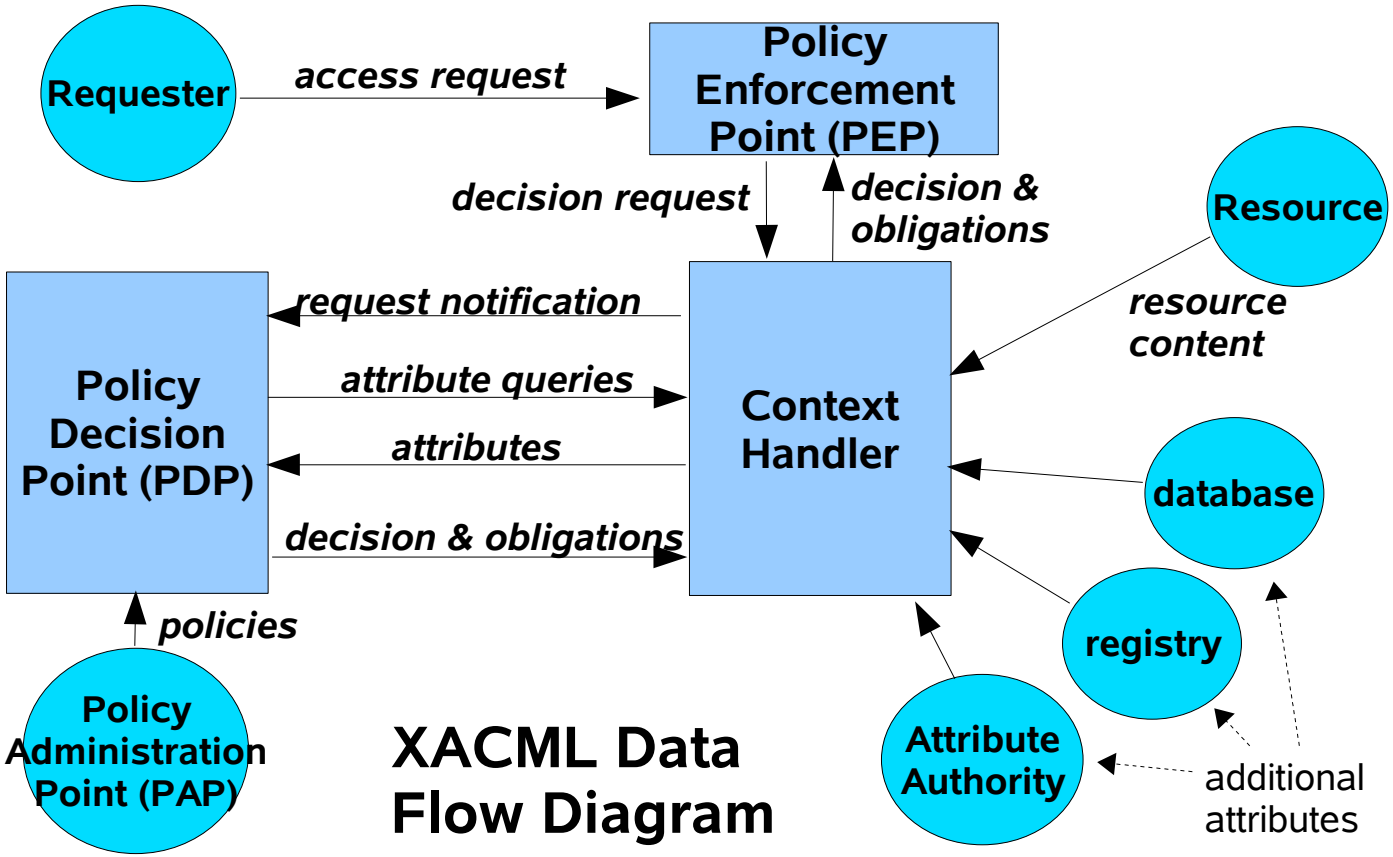
XACML Overview: adoption

- Used by more than 35 companies and architectures
  - Sun, BEA, IBM, Globus, Layer 7, Entrust, ...
  - ASTM Privilege Management Architecture (VHA)
  - Net-centric Enterprise Services Architecture (DISA)
  - Defense Readiness Reporting System (DoD)
  - Enterprise Dynamic Access Control (U.S. Navy)
- Publicly available implementations and tools, including unencumbered open source
- Over 135 published papers and articles so far

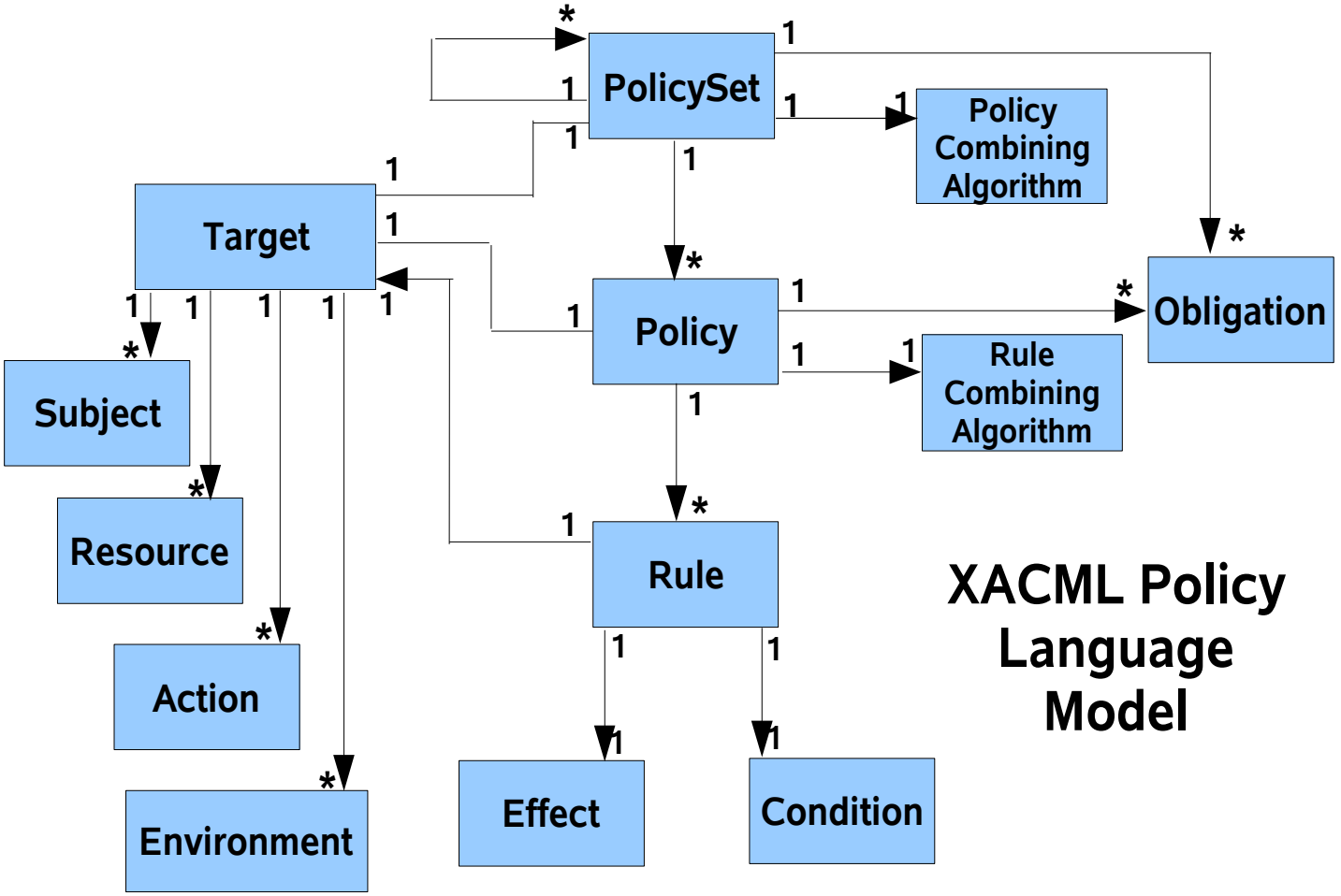
Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

How Does It Work: Data Flow



How Does It Work: Language Model

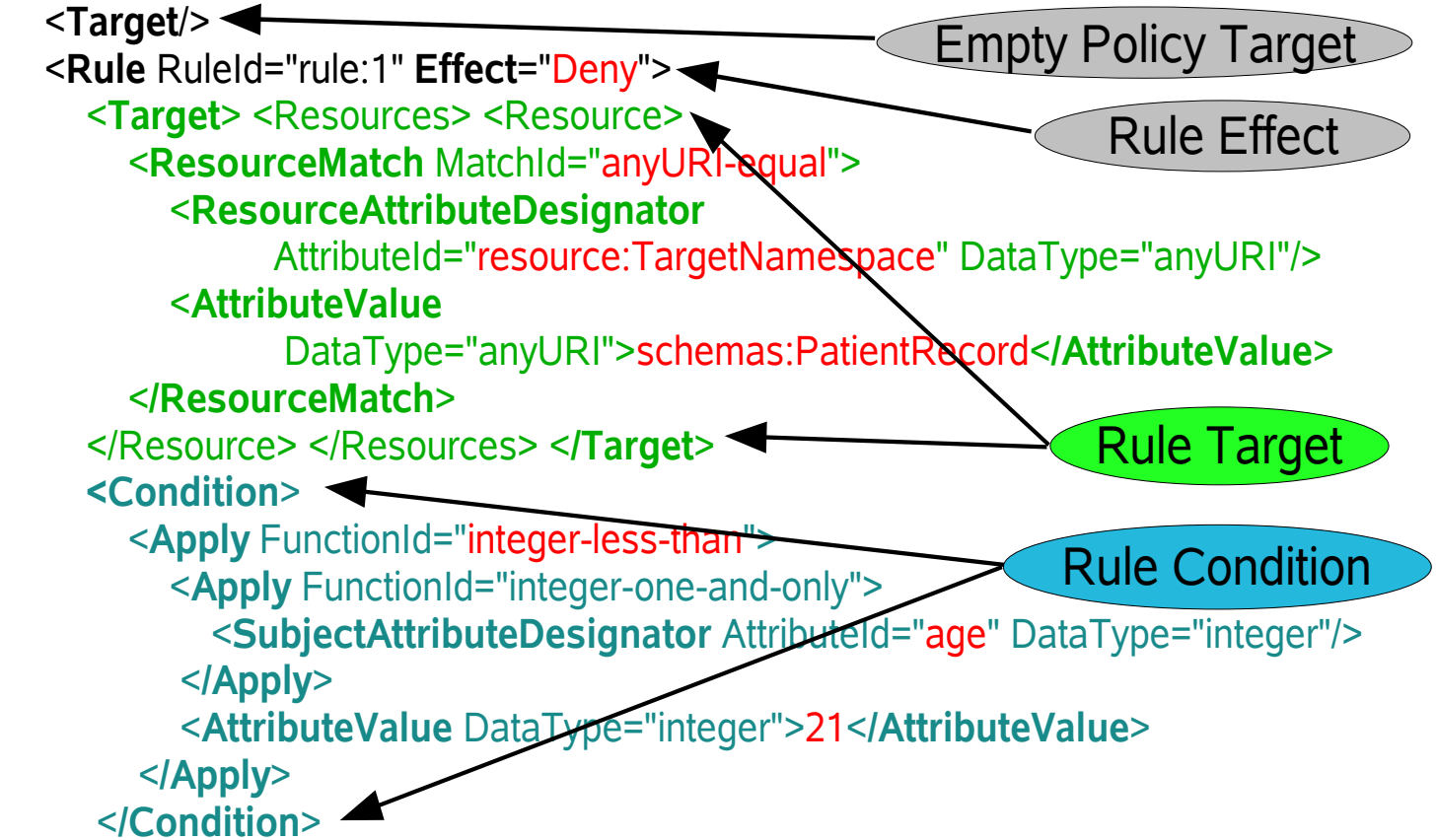


How Does It Work: Example Policy

"In order to access a patient record, you must be at least 21 years old."

How Does It Work: Example Policy

```
<Policy PolicyId="policy:1" RuleCombiningAlgorithm="deny-overrides">
  <Target/>
  <Rule RuleId="rule:1" Effect="Deny">
    <Target> <Resources> <Resource>
      <ResourceMatch MatchId="anyURI-equal">
        <ResourceAttributeDesignator
              AttributeId="resource:TargetNamespace" DataType="anyURI"/>
        <AttributeValue
              DataType="anyURI">schemas:PatientRecord</AttributeValue>
      </ResourceMatch>
    </Resource> </Resources> </Target>
    <Condition>
      <Apply FunctionId="integer-less-than">
        <Apply FunctionId="integer-one-and-only">
          <SubjectAttributeDesignator AttributeId="age" DataType="integer"/>
        </Apply>
        <AttributeValue DataType="integer">21</AttributeValue>
      </Apply>
    </Condition>
  </Rule>
</Policy>
```



Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

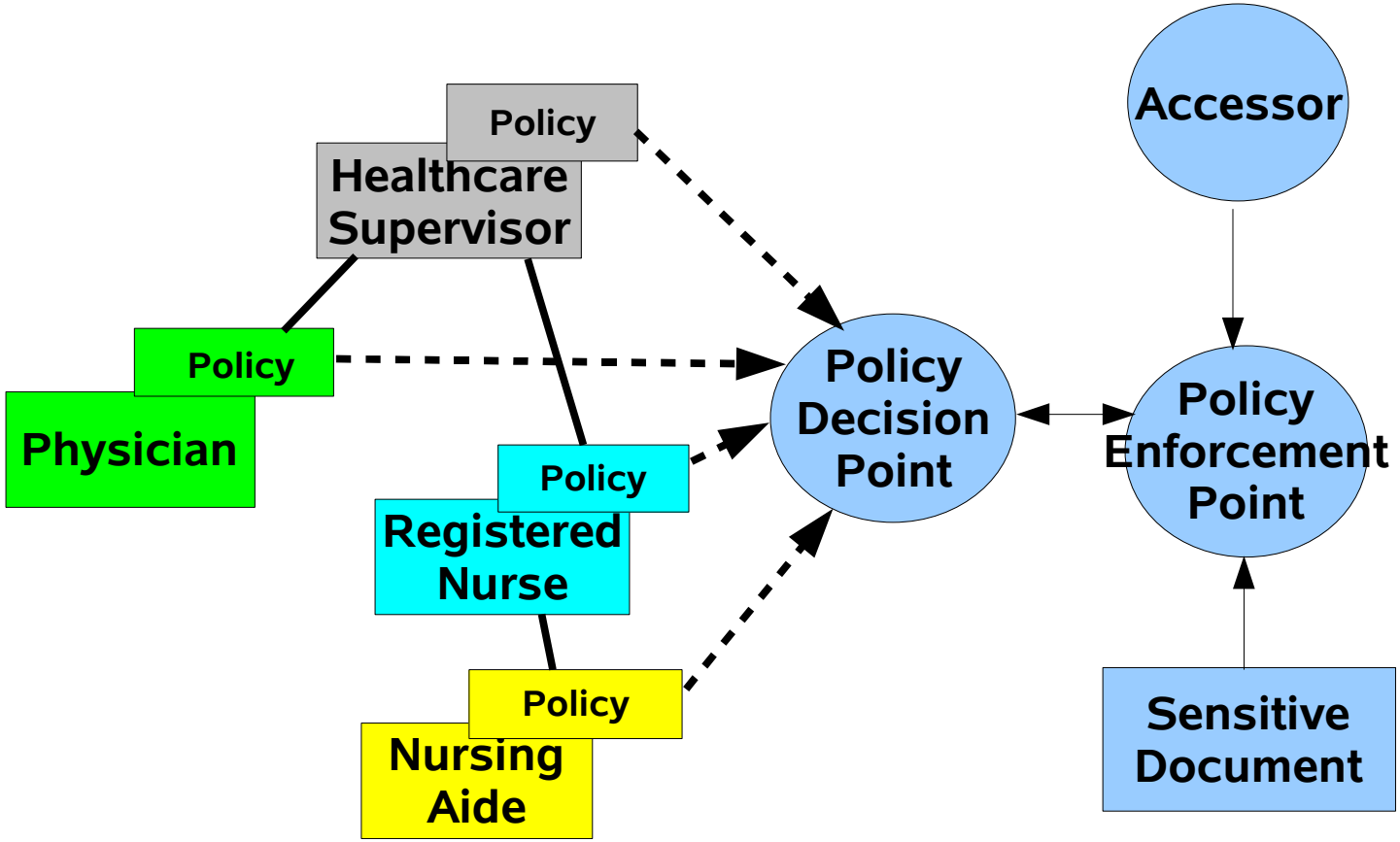
What Is Special About XACML?

- Domain-independent
- Protection of XML documents
- Distributed policies
- Optimized indexing of policies
- Rich set of standard functions and datatypes
- Profiles (see next slides)
- Policy administration and delegation coming

XACML Profiles: Privacy

- Standard XACML 2.0 +
- “Purpose” Attributes
  - Purpose for collecting information
  - Purpose for accessing information

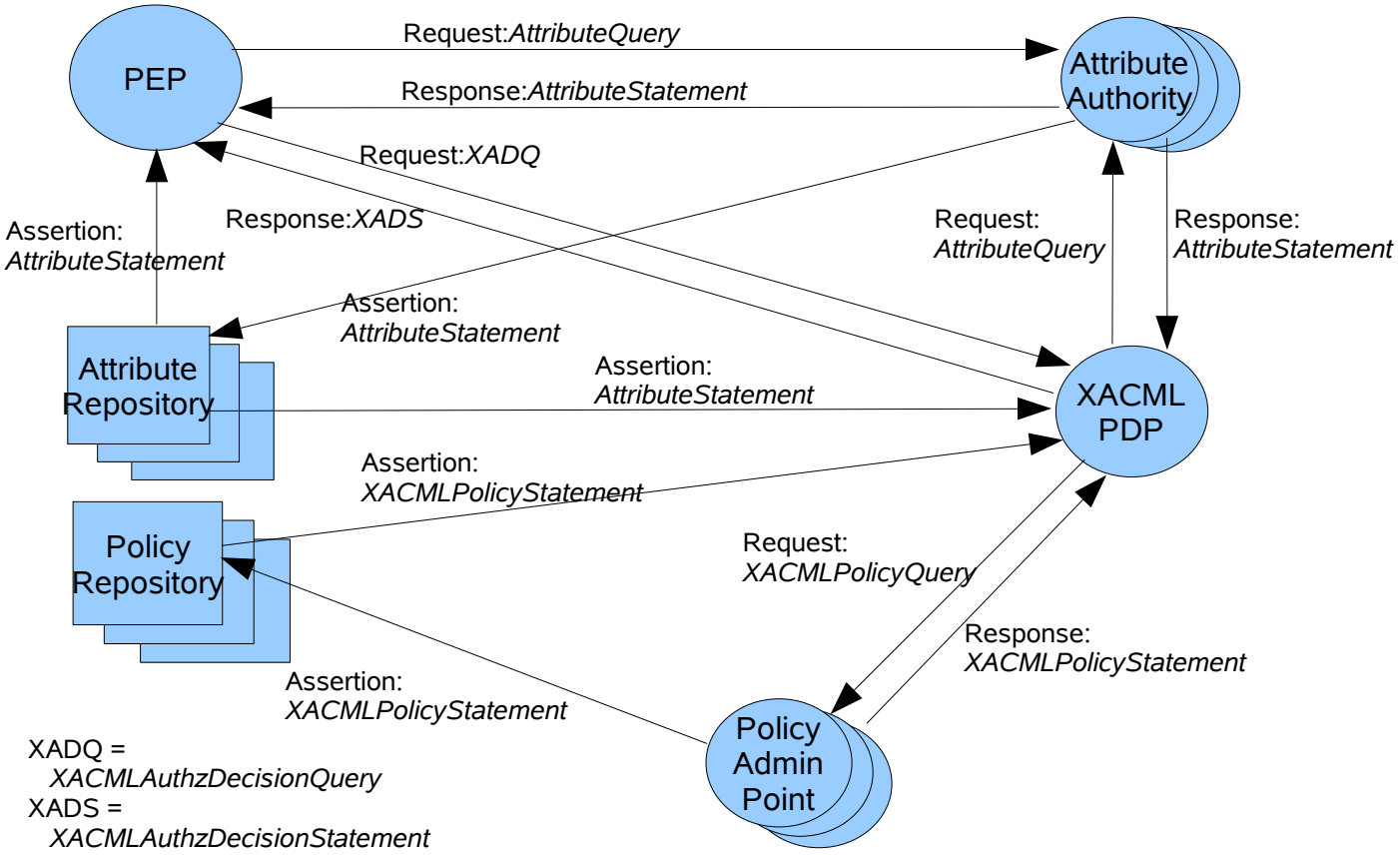
XACML Profiles:
Role Based Access Control



XACML Profiles:
Role Based Access Control

- ANSI/INCITS 359:2004:Information Technology – Role Based Access Control (RBAC)
- Developed with ANSI Standard developers

XACML Profiles: OASIS Security Assertion Markup Language (SAML)



Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

Comparison to Other Languages

- W3C's Platform for Privacy Preferences (P3P)
  - User-level, not enforcement level
  - No way to enforce the policies
- ISO/IEC 21000-5: ISO-REL
  - Designed for Digital Rights Management (media)
  - Not designed for enterprise-wide policies
- IBM's Enterprise Privacy Authorization Language (EPAL)
  - Proprietary; not in any standards organization
  - Largely a subset of XACML

Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

Further Information

- OASIS XACML Technical Committee Home Page
  - > http://www.oasis-open.org/committees/xacml
- XACML bibliography and list of deployments
  - > http://docs.oasis-open.org/xacml/xacmlRefs.html
- Sun's XACML Open Source Implementation
  - > http://sunxacml.sourceforge.net/
- More
  - > http://research.sun.com/projects/xacml

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

OASIS eXtensible Access Control Markup Language (XACML)

Anne Anderson
Anne.Anderson@sun.com

OASIS eXtensible Access Control Markup Language (XACML)

XML Community of Practice, 21 June 2006

Anne Anderson
Senior Staff Engineer
Sun Microsystems Laboratories

Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

Motivation for XACML

- Data more accessible today
- Regulatory requirements
- Protection at the enforcement level
- Protection of XML documents
- Standard language
  - > Interoperability
  - > Common policies, tools, analysis
  - > Auditability

Why is the eXtensible Access Control Markup Language important?

- Data is more easily accessible today, by more people, so control of access is much more important than in the past.

- Partly due to this, there are increasing regulatory requirements for the protection of data.

- In order to meet these requirements, a language for expressing access policies at the enforcement level is needed. A user-level privacy policy may say "personal information must be accessed only by authorized personnel", but at the enforcement level, it is necessary to define exactly which resources are "personal information" and exactly who is an "authorized person.", We need to control access by roles, by time of day, by document classification level, by subject's age, purpose of access, type of access, etc. Access Control Lists (ACLs) are inadequate for use with policy variables that go beyond local subject id, group id, resource id, and fixed set of actions.

- As more and more documents are written using XML, the language needs to support different protections on different parts of an XML document. For example, in a patient's medical record, the hospital accounting dept. staff should see the patient's account balance, but not the patient's diagnosis.

- Why not let vendors duke it out to produce good solutions for these problems?

o Policies need to be supported on multiple platforms. You don't want to have to rewrite each policy for every application that gives access to a resource: you want one policy that can be used by Java™ applications, web portal applications, the database system, the file system, etc. Right now, such a policy must usually be rewritten for each type of platform because there is no standard.

o With a standard language, a community of practice (!) develops around its use. This community can share best practices, policy templates, policy authoring and analysis tools, implementations, etc.

o Finally, with regulatory requirements comes a need to audit policies and accesses. With multiple languages in use, this task becomes a nightmare!

Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

XACML Overview

- XML-based
- Schemas for
  - > Policies
  - > Access decision requests
  - > Access decisions
- Rich set of functions and datatypes for expressing policies
- Specification: how to evaluate policies against access decision requests

XACML Overview: standardization

- OASIS Standard
- XACML 1.0 approved Feb. 2003
- XACML 2.0 approved Feb. 2005, including Profiles:
  - > Privacy
  - > Role Based Access Control
  - > OASIS Security Assertions Markup Language (SAML)
  - > ...
- XACML 3.0 in progress
  - > Policy-controlled policy administration

XACML Overview: adoption

- Used by more than 35 companies and architectures
  - > Sun, BEA, IBM, Globus, Layer 7, Entrust, ...
  - > ASTM Privilege Management Architecture (VHA)
  - > Net-centric Enterprise Services Architecture (DISA)
  - > Defense Readiness Reporting System (DoD)
  - > Enterprise Dynamic Access Control (U.S. Navy)
- Publicly available implementations and tools, including unencumbered open source
- Over 135 published papers and articles so far

XACML is currently used by more than 35 companies and architectures, and the list is growing steadily. As one example, XACML is already used in three Sun Microsystems products, and Sun's Access Manager product is migrating to XACML as part of becoming open source. Many other major companies are currently using XACML in products. In addition, a number of architectures, including the government architectures shown here, specify use of XACML.

It is impossible to provide a complete count of the organizations that use XACML. Most enterprises do not want to advertise their internal access control mechanisms, so do not make public the fact that they are using XACML. We know of some of these, and we suspect there are more.

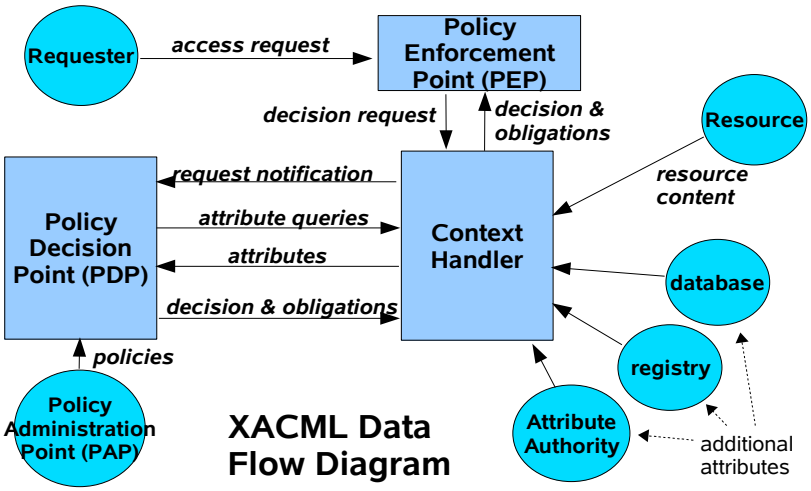
There are several publicly available implementations of XACML, and some publicly available tools. Some of these are unencumbered open source. There have been more than 135 articles and research papers published on XACML so far.

[A URL to the public list is included on the last slide]

Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

How Does It Work: Data Flow

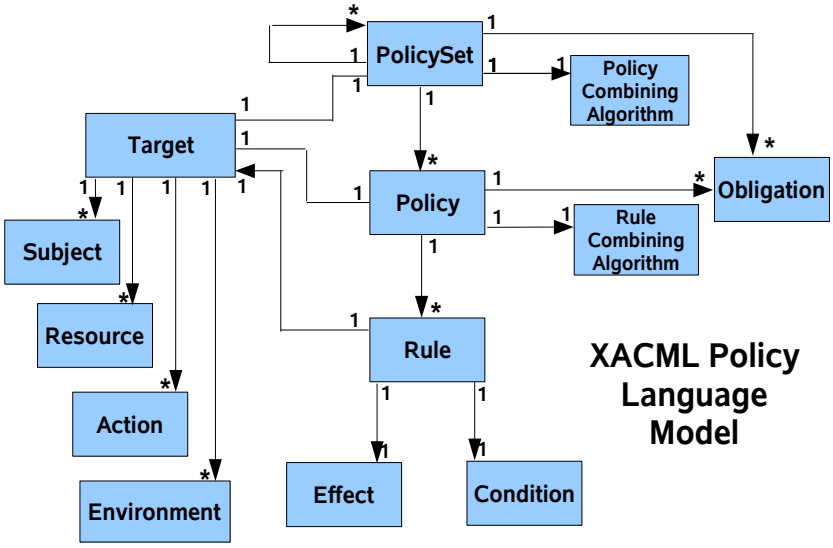


XACML is built around the logical separation of (usually) application-specific "Policy Enforcement Points", or "PEP", from a "Policy Decision Point", or "PDP". This has been a standard part of access control since ITU-T Recommendation X.812 (1995) | ISO/IEC 10181-3:1996, Information technology - Open Systems Interconnection - Security frameworks for open systems: **Access control framework (where they were called "Access Enforcement Function", or "AEF", and "Access Decision Function" or "ADF").**

A Policy Enforcement Point, or "monitor" is responsible for intercepting all attempts to access a resource. Since resource access is application specific, the PEP is also application-specific. The PEP collects information about the access request – who is making it, what resource is being accessed, what action is to be taken, etc. - and sends a "decision request" to a "Policy Decision Point". In XACML, the PDP is divided into two logical components. The "Context Handler" is implementation-specific. It knows how to retrieve the policy variables ("Attributes") used in that deployment. The PDP proper is completely standard – it evaluates policies against the information in the decision request, asking the Context Handler for additional Attributes as needed, and renders a decision of "Permit", "Deny", "Not Applicable", or "Indeterminate". "Not Applicable" means that no policy used by this PDP applied to the access request. "Indeterminate" means some error occurred that prevents the PDP from knowing what the correct response should be. Policies can contain "Obligations", or actions that must be performed as part of handling an access request, such as "Permit this access, but log it in the security log", or "Deny this access and notify the system administrator".

XACML specifies the XML format for policies and exactly how a PDP evaluates them. It specifies an abstract format for decision requests and decisions, which can optionally be (and often are) used as concrete request and response formats. XACML does not specify how the Context Handler obtains Attributes or authenticates them. It also does not specify how policies are written, stored, retrieved, or authenticated. Profiles help.

How Does It Work: Language Model



This is the XACML language model. Starting at the center, a “Policy” is made up of “Rules”. Policies can be grouped into “PolicySets”, which can also contain other “PolicySets”, creating a hierarchy. A top-level PolicySet can be viewed as a “tree” of descendant PolicySets, Policies, and Rules. From here on, when I say “Policy”, I mean “PolicySet” or “Policy”.

Each Policy and Rule has a “Target”. The Target is a simple predicate that specifies which Subjects, Resources, Actions, and Environments the Policy or Rule applies to. For example, a Target may specify that the Policy or Rule applies to every subject whose X.509 Distinguished Name contains “O=Sun Microsystems, C=US”.

An XACML PDP starts evaluating a decision request with a top-level Policy. It first evaluates that Policy's Target. If the Target is “false” the Policy is “Not Applicable”, and no further evaluation is done of either that Policy or of its descendants. If the Target is “true”, then the PDP evaluates the Policies or Rules at the next lower level, starting with their Targets. Again, if a Policy or Rule's Target is “false”, then that particular Policy or Rule is not evaluated further.

At the bottom level, if the Target of a Rule is “true”, then the Rule's “Condition” is evaluated. A “Condition” is an arbitrary Boolean combination of predicates. XACML supplies a rich collection of functions, including arithmetic, to use in forming predicates. Each “Rule” has an “Effect”, which is either “Permit” or “Deny”. If the Condition is “True”, then the Rule's “Effect” is returned. If the Condition is False, then “Not Applicable” is returned.

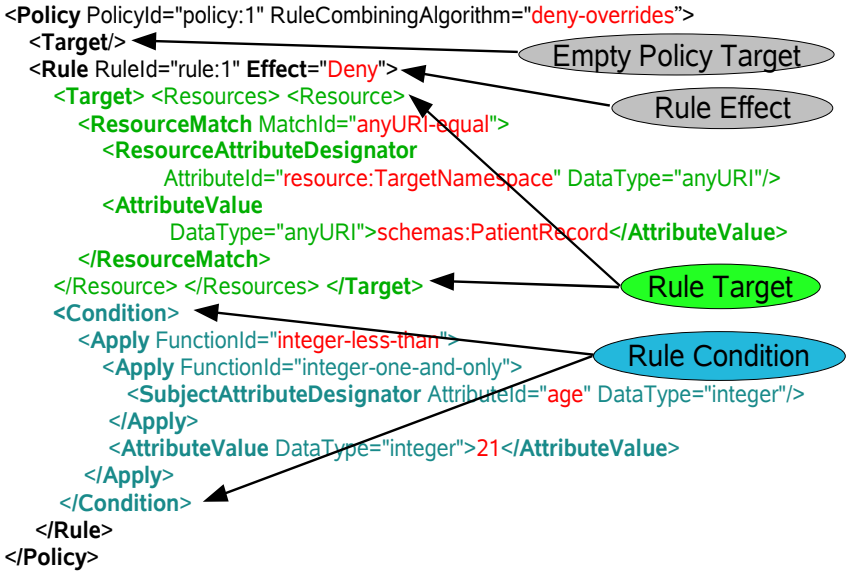
Each Policy specifies a “Combining Algorithm” that says what to do with the results from the Policies and Rules it contains. One such algorithm is “Deny Overrides”; in this case, if any child Policy or Rule evaluates to “Deny”, then the Policy evaluates to “Deny”. Another is “Permit Overrides”; in this case, if any child Policy or Rule evaluates to “Permit”, then the Policy evaluates to “Permit”.

Policies can have “Obligations” associated with them. The “Combining Algorithms” also tell how to combine the Obligations from different Policies.

How Does It Work: Example Policy

"In order to access a patient record, you must be at least 21 years old."

How Does It Work: Example Policy



This is the corresponding XACML policy, with abbreviated URIs and without XML namespace attributes. I won't try to explain it in detail, but here are the essential elements.

- The Target of this particular Policy is empty, meaning the Policy applies to all requests.

- The Policy contains one Rule, whose Effect is "Deny". The Target of the Rule says that the Rule applies to any access where the target namespace of the Resource being accessed is the PatientRecord schema. If this Target is "true", then the Condition is tested.

- The Rule will evaluate to Deny IF the resource being accessed is a PatientRecord instance AND if the person's age is less than 21.

- The RuleCombiningAlgorithm is "deny-overrides", so, if the Rule does evaluate to "Deny", then the Policy will return "Deny".

This example shows only a very small subset of XACML's capabilities, butl hope it gives you a sense of how the language works.

There is a large set of standard functions to use in specifying Conditions, and a large set of standard datatypes. The identities of Attributes are application-dependent – XACML doesn't care what they mean, only what their identifier is and what their DataType is.

Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

Next I want to describe a few of the special features of XACML.

What Is Special About XACML?

- Domain-independent
- Protection of XML documents
- Distributed policies
- Optimized indexing of policies
- Rich set of standard functions and datatypes
- Profiles (see next slides)
- Policy administration and delegation coming

What do I mean when I say XACML is "domain-independent"? It means each enterprise, organization, application, etc. that wants to use XACML decides on the set of policy variables it wants to use and what they mean. It is up to the domain to define identifiers and datatypes for these variables, and then to construct the policies in terms of these variables. XACML will work with any set of variables. This allows XACML to refer to resources, subjects, and actions using the terms and descriptors in use in the enterprise itself.

-XACML has special support for protection of XML documents. An Attribute can be an XPath expression instead of a named variable. In this case, the value of the Attribute is the value of the node (or nodes) selected by the XPath expression. This allows you to create various useful types of conditions. For example, you can let a Subject access a Patient Record if the Subject's name matches the "PatientName" field in the record. You can also give Subjects who are in the "Accounting Dept" access only to the "Patient Account" portion of a Patient Record and Subjects who are "Medical Personnel" access only to the "Medical History" portion of the record. And so on.

-XACML supports distributed policy management. For example, a PolicySet can include other policies by reference, instead of physically incorporating them. This allows a PolicySet to apply policies from the Legal Dept., the Finance Dept., and the Human Resources Dept., to each request, for example, without having to maintain them separately. XACML's " combining algorithms" also help with handling distributed policies.

-XACML's Targets are designed to allow efficient indexing of policies.

-XACML includes a rich set of arithmetic functions, comparison functions, string manipulation functions, regular expression functions, etc., as well as a large set of common data types. Implementations can extend these further if necessary.

-XACML has been profiled for several specific types of use cases. I'll talk about three of these profiles on the next few slides.

-XACML 3.0 will be adding the ability to specify policies that control who can create other policies. This supports both policy administration and delegation.

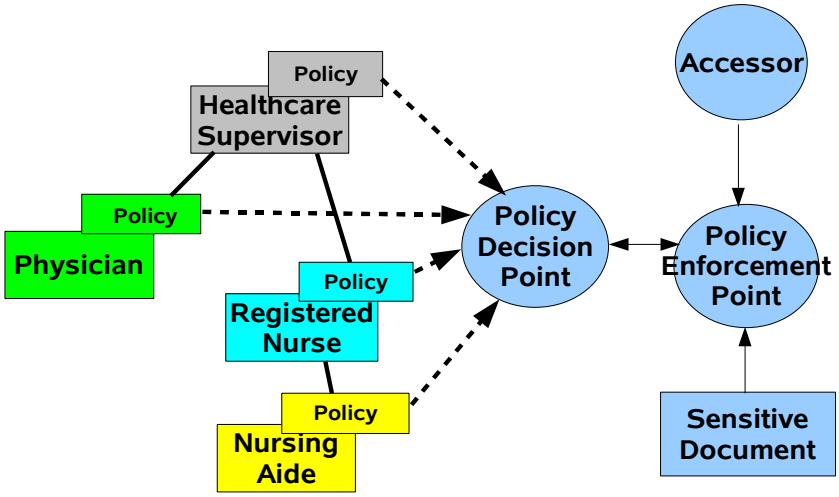
XACML Profiles: Privacy

- Standard XACML 2.0 +
- "Purpose" Attributes
  - > Purpose for collecting information
  - > Purpose for accessing information

XACML was designed from the start to support privacy protection. At the enforcement level, privacy protection must be integrated with access control because otherwise the two types of policies can conflict with each other – you can't have a privacy policy that says a Patient must always be allowed to see the Patient's own Medical Record if the access control policy says only employees of the hospital are allowed to see Medical Records.

One addition to XACML was needed to better support privacy policies. Privacy policies, unlike most traditional access control policies, often require that the purpose for which information is access must match the purpose for which the information was gathered. The XACML Privacy Profile defines standard attributes for describing the purposes for which information was gathered and the purposes for which information is being accessed. Policies can require that these two values match. The XACML Privacy Profile describes how to use these attributes.

XACML Profiles:
Role Based Access Control



Another important XACML profile is for "Role Based Access Control". This is where you want to organize the permissions in your organization around functional roles. In this diagram, a health care organization has four roles: Healthcare Supervisor, Physician, Registered Nurse, and Nursing Aide. There is a policy associated with each role that determines what a person associated with that role can do. XACML supports this organization of permissions around roles.

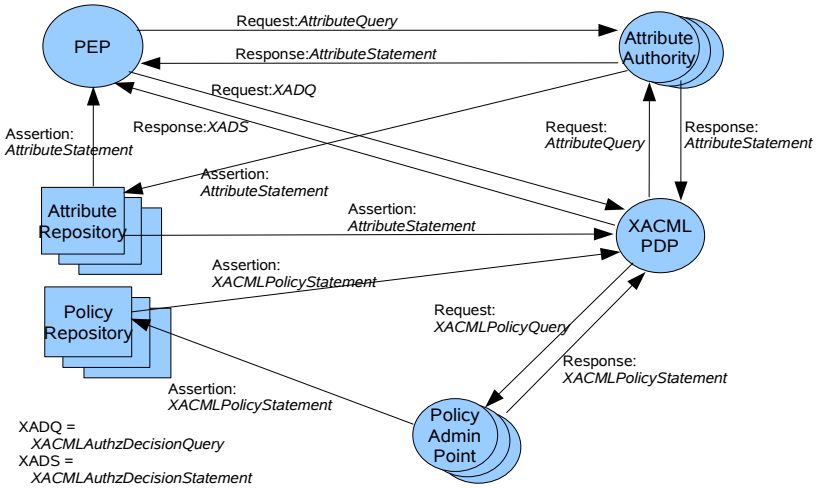
Some organizations structure their roles as a hierarchy. In this example, a "Healthcare Supervisor" should have all the permissions of either a Physician or a Registered Nurse, and a Registered Nurse should have all the permissions of a Nursing Aide. XACML supports this hierarchical organization of roles also.

XACML Profiles:
Role Based Access Control

- ANSI/INCITS 359:2004:Information Technology – Role Based Access Control (RBAC)
- Developed with ANSI Standard developers

XACML's Role Based Access Control model is based on the ANSI/INCITS 359 Standard for Role Based Access Control approved in 2004. XACML's support for Role Based Access Control was developed in conjunction with the developers of the ANSI standard.

XACML Profiles: OASIS Security Assertion Markup Language (SAML)



Another XACML profile supports use of the OASIS Standard Security Assertion Markup Language, or "SAML", to implement communication between the various entities in the XACML model. The profile allows SAML to be used to request authorization decisions, policies, and attributes, and to protect these communications using digital signatures. XACML's profile was developed with help from the SAML developers.

Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

Next I will talk a little about three other languages related to access control and privacy.

Comparison to Other Languages

- W3C's Platform for Privacy Preferences (P3P)
  - > User-level, not enforcement level
  - > No way to enforce the policies
- ISO/IEC 21000-5: ISO-REL
  - > Designed for Digital Rights Management (media)
  - > Not designed for enterprise-wide policies
- IBM's Enterprise Privacy Authorization Language (EPAL)
  - > Proprietary; not in any standards organization
  - > Largely a subset of XACML

- W3C's Platform for Privacy Preferences or "P3P" is a W3C Recommendation; a new version is in last call now. P3P deals only with user-level policies, and only in generic terms. It is not designed to specify how these policies apply to particular resources in the enterprise. It is not linked to policy enforcement mechanisms.

- ISO/IEC 21000-5:ISO-REL (or MPEG-REL) was standardized for use with MPEG-encoded media. It is based on the eXtensible Rights Markup Language (XrML) from ContentGuard, and a license is required to use it – it is not royalty-free. ISO-REL is not designed for access control or privacy, but for "digital rights management". It is not designed for the expression of enterprise-wide policies, but for domain-specific rights associated with particular media copies.

- IBM's Enterprise Privacy Authorization Language is not a standard, and is not being worked on in any standards organization. It is a proprietary IBM language. In functionality, it is largely a subset of XACML (see comparison at http://research.sun.com/projects/xacml). EPAL is not suitable for enterprise-wide policies; it quickly becomes unmanageable as more rules are added. It was also not designed as a general-purpose access control language. This means that an enterprise would have to write its access control policies using a different language, and these two sets of policies could conflict.

Outline

- Motivation
- XACML Overview
- How Does It Work?
- What's Special About XACML?
- Comparison to Other Languages
- Further Information

Further Information

- OASIS XACML Technical Committee Home Page
  - > http://www.oasis-open.org/committees/xacml
- XACML bibliography and list of deployments
  - > http://docs.oasis-open.org/xacml/xacmlRefs.html
- Sun's XACML Open Source Implementation
  - > http://sunxacml.sourceforge.net/
- More
  - > http://research.sun.com/projects/xacml

The OASIS XACML Technical Committee's Home Page has links to all the schemas and specifications, including all the profiles. It also contains links to all known publicly available implementations of XACML (4 so far), as well as lots of other information.

The XACML bibliography contains a list of all papers and articles published about XACML, and a list of all known products and architectures that use it. It is updated every few months.

Sun's XACML Open Source Implementation is unencumbered; it is available under a royalty-free, non-viral BSD license. This supports XACML 1.0 and 1.1 completely, and supports almost all of XACML 2.0.

There is more information related to XACML at the research.sun.com page shown.

This concludes my presentation.

OASIS eXtensible Access Control Markup Language (XACML)

Anne Anderson
Anne.Anderson@sun.com