Clinical Record Use Cases

Title:	Clinical Record Use Cases
Terse Description:	Control of the creation, maintenance, and access of medical records and messages coded in XML.
Version:	v0.1
Submitted by:	Fred Moses
Date:	September 4, 2001

Summary

Access to medical records is governed by ethical and legal privacy requirements and the preferences of the patient. This use case and its variants illustrate related confidentiality needs.

Scope

Medical record creation, storage, access, and messaging system and its users.

Actors

- Creators and readers of medical documents such as physicians and other health care givers
- Patients
- Those associated with Patients who have access privileges
- Payers
- Institutions (HMOs, government bodies) permitted access.

Assumptions

- The Health Level Seven Clinical Document Architecture and usage drawn from discussions about it form reasonable models for the creation, management, and accessing of medical information about individual patients. The interpretation of the HL7 standards is strictly that of this writer.

Non-technical Factors

HIPAA and other privacy legislation, medical ethics.

Process Sequence

Flow diagrams are not provided in this version.

Primary Process Flow

A physician creates a record with administrative, medical, and privacy content, signs it, and has it stored (in XML format) in a records system.

Key Points:

- Other personnel may collect portions of the record, such as the administrative information.
- Access policy must have granularity at the level of elements within the document and individuals within the actor population.
- Access policy must be included with the document.
- The document must have a nonreputable signature.
- Once signed, the document itself may not be modified.

Alpha Process Variant: Record retrieval

A physician or other permitted actor retrieves all or parts of a record for review or transmission to other parties.

Key Points:

- The portion of a document that may be retrieved depends upon the requestor and privacy conditions included in document. For example:
  - Patient restricts access to specific administrative info (address and phone number) to prevent abusive ex-spouse from finding her.
  - Restrictions extend beyond the originating organization and follow the record or message to another. (This may warrant the encryption of restricted portions.)
  - Differential access restrictions for especially private information such as psych notes. That is, while most of a record may be made available to a new actor, restrictions may be applied in the process.

Beta Process Variant: Record transmittal

A permitted actor retrieves all or part of a record for transmission to other parties. They must be bound by the same restrictions that already apply to the information.

Key Points:

- Restrictions extend beyond the originating organization. Encryption may be a means of enforcing this.
- Necessary agreements between originator and receiver are beyond the scope of this use case.

Gamma Process Variant: Record addendum

A physician or other caregiver creates an addendum to a record with administrative, medical, and/or privacy content, signs it, and has it stored (in XML format) with the existing record in the records system.

Key Points:

- Since signed records or portions of them may not be modified, some form of a attachment or addendum must be used.
- Changes in access permissions may affect the previously existing document and any addenda. Both patient and care giver can add restrictions. Only the patient can cause restrictions to be removed. (See the cases regarding information withheld from the patient, below.)
- Any addenda to the access policy must be included with the document.
- Should a form of version control be applied?

- The result must have a nonreputable signature.
- Once signed, the addendum itself may not be modified.

Delta Process Variant: "Breaking the glass"

A patient arrives at the emergency room unconscious. Caregiver(s) need to be able to assume special privileges in order to gain access to information that was restricted, but may be critical in the patient's care.

Key Points:

- There need to be people, possibly outside the normal flow, who have special privileges.
  - Do they need to possess a special decryption key?
  - Do there need to be multiple decryption keys such that no single person can break glass.
- When extraordinary measures are invoked, should a standard mechanism attach a note to the record? See the comment regarding version control, above.

Epsilon Process Variant: Information is withheld from patient

A psychiatrist receives information that s/he believes could be harmful to the patient or others if disclosed to the patient. In accordance with the law in the patient's state, the psychiatrist marks this information as not to be disclosed to patient. The patient requests access to his/her psychiatric records. Access to the restricted documents is denied.

Key Points:

- The patient isn't the legal owner of his/her records. Except in legally identified cases such as this, however, the patient has the right to see his/her own record. Thus, there is a policy that circumstances may modify.

Zeta Process Variant: Patient overrides restrictions

The patient in the previous example obtains an override of the restrictions through legal recourse. Access is permitted.

Key Points:

- Legal maneuvers are outside the scope of this use case.
- There is a need for attaching new access privileges to an existing document.

Glossary

Caregiver
Physician, nurse, or other person providing health care. The HIPAA rules gives strict definitions for this and other personages and devices associated with the health care process. These are outside the scope of this use case. An informal meaning will suffice.

HIPAA
Health Insurance Portability and Accountability Act of 1996 - An act of Congress specifying, among other things, privacy standards for medical records. This is augmented by Department of Health and Human Services rules. See the Web site given below.

Nonreputable signature

A signature signed in such a fashion that the signer couldn't refute it. See, for example, the XMLD specification for which there is a link below.

References

Health Level Seven - http://www.hl7.org/

- Structured Documents Technical Committee
- XML Special Interest Group

Modeling and Methodology HIPAA - http://www.hcfa.gov/hipaa/hipaahm.htm

XML - Signature Syntax and Processing - http://www.w3.org/TR/xmldsig-core/

ebXMLRegistry-Restricting Read-Write Access

Title:	ebXMLRegistry-Restricting Read-Write Access
Terse Description:	Limiting read-write (read, approve, deprecate, remove) access for the Registry contents to specified subjects.
Version:	V 0.5
Submitted By:	Suresh Damodaran
Date:	Sept 4, 2001

Summary

Scope

Actors:

- Registered User: Affiliated with either the Submitting Organization or Partner Organization.
- Registry Guest: Is not affiliated with either the Submitting Organization or Partner Organizations.
- Submitting Organization: Who submits RegistryObject
- Partner Organization: Partners of submitting organization

Assumptions

It is assumed that the information on Registered Users affiliated with a Partner or Submitting Organization is available in the Registry. Registered User and Registry Guest are authenticated.

Non-Technical Factors

Process Sequence

Primary Process

A Submitting Organization (SO) submits a RegistryObject to a Registry. SO also submits to Registry an AccessControlPolicy associated with the RegistryObject. This AccessControlPolicy allows only selected Users of SO or Partner Organizations to have read, approve, deprecate, and remove access of the RegistryObject. All objects in the registry have a unique id specified by *Universally Unique Identifier (UUID)* and must conform to the format of a URN that specifies a DCE 128 bit UUID as specified in UUID [ebRS:Section 7.3.1, UUID].The Registered Users affiliated with Partner Organizations or Submitting Organization may be specified in the AccessControlPolicy using Identity, Role, or Group information.

Flow Diagram

Key Points

Glossary

References

[ebRS] ebXML Registry Services Specification - http://www.ebxml.org/specs/ebRS.pdf

[ebRIM] ebXML Registry Information Model 1.0 - http://www.ebxml.org/specs/ebRIM.pdf

[UUID] DCE 128 bit Universal Unique Identifier

- http://www.opengroup.org/onlinepubs/009629399/apdxa.htm#tagcjh_20
- http://www.opengroup.org/publications/catalog/c706.htm
- http://www.w3.org/TR/REC-xml

Online Access Control

Title: Online Access Control

Terse Description: Policy determines if access should be allowed to online resources

Version: 1.0

Submitted By: Hal Lockhart

Date: September 4, 2001

Summary

A user or process in an online environment makes a request of an online server. A policy is evaluated to determine if the access should be allowed. Elements within the server act as a Policy Enforcement Point, either allowing or denying access.

Scope

The scope includes any online server application environment, such as HTTP; Java Applications, including Servlet, Java Server Pages and J2EE; and CORBA. It could also apply to other Internet protocols, such as ftp or pop3. It could apply to legacy environments, such as mainframe transaction processing. It could also apply to emerging environments, such as XML Protocol. The access control is typically non-discretionary, but many of the existing schemes are based on discretionary methods, e.g. ACLs.

Actors

- System Entity that originates the request,
- Server (PEP),
- PDP

Assumptions

Non-technical Factors

Many of these environments have existing access control schemes associated with them. However the existence of a number of third party Access Management products with capabilities not present in the existing schemes suggests that they do not completely meet user requirements. Furthermore, since distributed applications are often built with a combination of these technologies, the use of multiple schemes is both inconvenient and error prone.

Process Sequence

Primary Process Flow

1. System Entity makes application request to Server containing PEP
2. PEP requests policy decision from PDP specifying target (local or remote)
3. PDP locates all applicable policies

4. PDP obtains necessary policy inputs from PIP (local or remote)
5. PDP evaluates policy to determine if access should be allowed
6. PDP informs PEP of decision
7. PEP permits action or returns error
8. [Optional] PDP makes determination to record information in Audit trail base on same or different inputs

Targets

The target of a request depends on the environment. In a Web environment it is an HTTP or HTTPS URL or the path component of the URL. This may be qualified by the HTTP operation specified, however this may be omitted because it is not possible in general to determine what the semantic of the particular request may be, e.g. Read or Write. In a remote invocation environment, the request typically specifies a method on an object. However, EJB security makes it possible to distinguish among different signatures on the same method. There is also utility to providing for targets that are arbitrary strings that may be meaningful to an application.

Conditions

The decision to allow access may be based on any or all of the following criteria.

- User possess a specified attribute (member of organization)
- User possesses a specified attribute with a specified value (member of Admin group)
- User possesses a specified numeric attribute that matches a numeric test against a constant (transaction limit > 1000)
- Current time is in specified range (between 9AM and 5PM)
- Current day of week is as specified (Saturday or Sunday)
- Client IP Address or DNS name is as specified
- Server IP Address or DNS name is as specified
- User authenticated using specified method (PKI)
- Connection is protected (TLS in use)

It should be possible to combine these conditions using the standard Boolean operators.

The normal consequence of policy evaluation is to allow or deny access. A policy decision may also be made to generate an Audit Trail record corresponding to the request. In this case, all the above criteria may be used and in addition:

Was the request allowed or refused Audit could be a provisional result of the decision, however this is inconvenient for two reasons:

- The final criterion mentioned applies to the audit decision and not to the authorization decision.
- It is frequently desired to enforce access control and not audit or generate audit records without checking access.

For both of these reasons it is simpler to have distinct Authorization and Audit Trail policies, instead of treating them as multiple consequences to a single policy.

Flow Diagram

Key Points

- A wide variety of resources can the target of the policy.

- Policy inputs include many other factors than subject attributes. In fact subject attributes may not be used at all in some decisions.
- The protocol used to make the request is irrelevant to the policy decision, except for its security properties

Alpha Process Variant

It is also possible to support lazy Authentication. This is an explicit part of the HTTP and Servlet protocols. In step 4, if the PDP determines that authenticated subject attributes are a required policy input and the user has not previously authenticated, he or she may be challenged to authenticate at that time.

Flow Diagram

Key Points

- Lazy Authentication

Beta Process Variant

Another variant occurs when the PDP recognizes that the policy evaluation failed because some factor that the requestor may be able to alter. Examples include:

- An insufficiently strong authentication method was used, or
- The communications channel is inadequately protected.

By signaling the problem to the application or the user, it may be possible to remedy the deficiency.

Even when user action is not required, it may be desirable for performance reasons to only gather certain inputs once it is known they are needed. For example, a reverse DNS lookup of the client's IP address may be omitted unless specifically required.

Flow Diagram

Key Points

- Detailed feedback of reasons for failed policy evaluation

Glossary

References

Policy Provisioning

Title: Policy Provisioning

Terse Description: Policies are distributed from PRPs to PDPs

Version: 1.0

Submitted By: Hal Lockhart

Date: September 4, 2001

Summary

Previously created or modified policies are transferred from a Policy Retrieval Point (PRP) to a Policy Decision Point.

Scope

The scope includes any environments where PDPs utilize policies made available from a PRP.

Actors

- PRP
- PDP

Assumptions

Non-technical Factors

Process Sequence

Primary Process Flow

In this use case, the PDP simply requests policies from the PRP. The PDP might initiate the request based on elapsed time since the last update or some other criterion.

Flow Diagram

Key Points

- A reliable protocol to upload policies.
- The type of policy representation is irrelevant.

Alpha Process Variant

In this case, the PRP notifies the PDP that new policies are available. The PDP can then request the policies as in the previous case.

There are two reasons for this scenario as compared to having the PRP push policies to the PDP.

1. The PDP may be resource constrained. This allows it to control when and how it updates its policies.
2. The second part of the protocol is exactly the same as the Simple Pull, thus simplifying specification, implementation and testing.

Flow Diagram

Key Points

- PDP is notified when policies have changed.
- PDP controls the transfer process.

Glossary

References

SAML Authorization Decision Request and Assertion

Title: SAML Authorization Decision Request and Assertion

Terse Description: Policy inputs are conveyed between a PEP and PDP or between a PDP and a PIP

Version: 1.0

Submitted By: Hal Lockhart

Date: September 4, 2001

Summary

A PEP formulates a SAML request for an Authorization Decision, by specifying the policy inputs that apply. A PDP replies with an assertion that also specifies the policy inputs applied to the decision.

The PDP may also request the necessary input values from a PIP, which in turn returns the values.

Scope

The scope includes any environments where SAML Authorization Decision Requests and Assertions are used or where a PIP is located remotely from a PDP.

Actors

- PEP
- PDP
- PIP

Assumptions

Non-technical Factors

Process Sequence

Primary Process Flow

1. A PEP requests a SAML Authorization Decision Assertion, specifying the policy inputs.
2. The PDP determines that it lacks some of the inputs required for policy evaluation. It requests additional data from the PIP.
3. The PIP replies with the necessary inputs.
4. The PDP evaluates the relevant policies and issues the Authorization Decision Assertion, specifying the policy inputs utilized.

Flow Diagram

Key Points

- A syntax to identify policy inputs and specify their values.

Attribute-dependent Access Control on XML Resources

Title:	Attribute-dependent Access Control on XML Resources
Terse Description:	Filtering online catalog XML containing different security categories
Version:	v1.0
Submitted By:	Michiharu Kudo (IBM)
Date:	September 3, 2001

Summary

This use case presents attribute-dependent access control policies using online catalog XML document. Access decisions defer dependent on the value of the specific attribute of the target document as well as time of the access.

Scope

Target resource is written in XML

Actors

Assumptions

- Target online catalog data is written in XML. (XML with/without schema definition.)
- Access control policies are represented as a set of triplet <object, subject, action, condition>.
- Access initiators are users who are categorized into two classes, premium member and normal member.
- XACML provides necessary and useful set of primitives for representing condition.
- Access control policies are defined using attribute values in the target resource and/or the associated security classes of the target document.
- In the policy rules, grant and denial permission are used simultaneously.

Non-technical Factors:

Process Sequence

Primary Process Flow: Filter online catalog

This is a typical online shopping application for cyber marketplaces. XML is used to store online catalog data that contains items for sell. There are two classes for buyers: normal members and premium members. The catalog includes all available items, including some that are available only to premium members. Selling information is labeled as “normal”, “premium”, or “all”. The access control policy says that the normal members cannot read any information for premium members, and the premium members cannot read any information for normal members.

The catalog XML document in this example contains two available items: “Digital Video Camera” and “Luxury Sofa”. The “Digital Video Camera” is sold for both normal and premium members. The selling period is from

10th Sep. 2001 to 17th Oct. 2001 and the price is US$489.99. The normal members have to pay US$39.99 as a shipping fee. The normal members get 1,000 bonus points but the premium members get 3,000 points. The "Luxury Sofa" is sold only for premium members. This is sold through the years 2001 and 2002 at the price of US$3,499.99. Original catalog XML document is as follows:

```
<?xml version="1.0"?>
<!DOCTYPE catalog SYSTEM "catalog_target.dtd">
<catalog>
  <item member="all">
    <name>Digital Video Camera</name>
    <period>
      <start_time>9/10/01 0:0 AM</start_time>
      <end_time>9/17/01 11:59 PM</end_time>
    </period>
    <price currency="USD">489.99</price>
    <ship_fee currency="USD" member="normal">39.99</ship_fee>
    <advantage>
      <point member="normal">1000</point>
      <point member="premium">3000</point>
    </advantage>
  </item>
  <item member="premium">
    <name>Luxury Sofa</name>
    <period>
      <start_time>1/1/01 0:0 AM</start_time>
      <end_time>12/31/02 11:59 PM</end_time>
    </period>
    <price currency="USD">3499.99</price>
  </item>
</catalog>
```

A set of access control policies is described as follows:

1. Normal members and premium members can read each item except for the period element if the selling period condition is satisfied.
2. Normal member cannot read any information where member attribute is premium.
3. Premium member cannot read any information where member attribute is normal.

Access request: *Can Normal read the root node of the catalog document?*

The PDP makes an access decision against the access request based on the access control policy rules. Decisions are the following:

Access decision: *Normal member can read <item> for Digital Camera and descendant nodes except for <advantage> for premium members and <period> element.*

The following XML shows a resultant requesters view who is a normal member.

```
<?xml version="1.0"?>
<!DOCTYPE catalog SYSTEM "catalog_target.dtd">
<catalog>
  <item member="all">
    <name>Digital Video Camera</name>
    <price currency="USD">489.99</price>
    <ship_fee currency="USD" member="normal">39.99</ship_fee>
    <advantage>
      <point member="normal">1000</point>
    </advantage>
  </item>
</catalog>
```

Key Points:

- 1st policy specifies temporal condition in the condition portion. If condition does not hold, the 1st policy will not be considered in making the decision. Since the decisions are determined in run-time, we can call it dynamic access control.
- 2nd and 3rd policy specifies denial access control policy rules. There are two ways to represent these rules:

  - The first is to write an element selection formula in the object pointers field, e.g. using Xpath, //*[@member='premium'] means any element whose member attribute is 'premium'. This is the efficient way of using Xpath for XML access control systems.
  - The second is to write a condition formula that checks whether or not the current node has premium attribute, e.g. getValue(./@member) = 'premium'.
- PDP has to know the value of the referred element or attribute if the policy rules use them. There are two ways to solve this:
  - Before calling PDP, PEP retrieves the target resource and forward whole target resource to PDP as well as the other access request parameters.
  - Before calling PDP, PEP retrieves the target resource but does not forward whole target resource but send only a needed portion. This implies PEP must understand the contents of the policy rules.
- In this case, conflicting decision can be derived from the rule 1 and 2. From the rule 1, the normal member can read information for premium members, while they are not allowed to read it from the rule 2. The conflict resolution policy is needed to solve this decision. Normally, denial-takes-precedence policy would be more appropriate.

Glossary

Access mode
Read, write, delete, and create

Access request
Data submitted by a requester that contains target XML resource, target node (element or attribute), access mode, and an authenticated user identity and its attributes.

Access response
Data returned by a PDP that contains access decision (grant or deny), target node, access mode, user information, and additional conditions (provisions, advise, etc.)

Attribute
Sub-structure of the target XML resource

Element
Sub-structure of the target XML resource

Target resource
The resource that consists of sub-structure such as element and attribute.

References:

[XACL1] XML Access Control Language (XACL) - http://www.trl.ibm.com/projects/xml/xacl/index.htm

[XACL2] XACL reference implementation - http://alphaworks.ibm.com/tech/xmlsecuritysuite

Appendix: XACL Sample Policy

```
<policy xmlns="http://www.trl.ibm.com/projects/xml/xacl">
<!-- ==================================================
   1. Normal members and premium members can read any
      items in the online catalog, if the selling period
      condition is satisfied.
  ================================================== -->
   <xacl>
     <object href="/catalog/item"/>
     <rule>
       <acl>
```

```
        <subject>
          <group>normal_member</group>
        </subject>
        <subject>
          <group>premium_member</group>
        </subject>
        <action name=”read” permission=”grant”/>
        <condition operation=”and”>
          <predicate name=”compareDate”>
            <parameter value=”after”/>
            <parameter><function name=”getDate”/></parameter>
            <parameter><function name=”getValue”>
              <parameter value=”./period/start_time”/></function></parameter>
          </predicate>
          <predicate name=”compareDate”>
            <parameter value=”before”/>
            <parameter><function name=”getDate”/></parameter>
            <parameter><function name=”getValue”>
              <parameter value=”./period/end_time”/></function></parameter>
          </predicate>
        </condition>
      </acl>
    </rule>
  </xacl>
<!-- ==================================================
  2. Normal member cannot read any information for
     premium members.
 ================================================== -->
  <xacl>
    <object href=”//*[@member=’premium’]”/>
    <rule>
      <acl>
        <subject>
          <group>normal_member</group>
        </subject>
        <action name=”read” permission=”deny”/>
      </acl>
    </rule>
  </xacl>
<!-- ==================================================
  3. Premium member cannot read any information for
     normal members.
 ================================================== -->
  <xacl>
    <object href=”//*[@member=’normal’]”/>
    <rule>
      <acl>
        <subject>
          <group>premium_member</group>
        </subject>
        <action name=”read” permission=”deny”/>
      </acl>
    </rule>
  </xacl>
</policy>
```

Requester-dependent Access Control on XML Resources

Title:	Requester-dependent Access Control on XML Resources
Terse Description:	Groupware application using XML as a central repository
Version:	[v1.0]
Submitted By:	Michiharu Kudo (IBM)
Date:	September 3, 2001

Summary

This use case presents requester-dependent access control policies using academic paper-reviewing system. Review summary document is represented in XML and an access (read and write) to the document is strictly controlled based on the requester's privilege.

Scope

Target resource is written in XML

Actors

Assumptions

- Target review summary document is written in XML. (XML with/without schema definition.)
- Access control policies are represented as a set of triplet <object, subject, action, condition>.
- Access initiators are users who play the following four roles, chair, committee, reviewer, and author.
- XACML provides necessary and useful set of primitives for representing condition.
- Access control policies are defined using attribute values in the target resource and/or the associated security classes of the target document.

Non-technical Factors:

Process Sequence

Overview

Groupware application for paper review

Primary Process Flow

This application simulates a typical review process for academic papers. This example illustrates how the XML access control is applied to applications that need information sharing and/or updating among multiple participants who play different roles. The review process can be described as follows:

1. Authors submit their papers to the submission server. A chairperson assigns one or more reviewers to each submitted paper.

2. The reviewers read the assigned paper and evaluate it.
3. The program committee members read the reviewers’ evaluations and decide whether or not each paper should be accepted.
4. The chairperson decides on the list of accepted papers.
5. The authors receive notifications of acceptance or rejection.

We simplify the above process and produce a review summary document. The summary document stores data such as author information and evaluation results. The following summary document includes paper submissions from authors Xerces, Stackman, and Dreamer. Each submission consists of <paper_title>, <paper_number>, <author>, <review>, <result>, and <confirmation> fields. The <paper_title>, <paper_number>, and <author> fields stores submission information. The <review> section is used by reviewers. The <result> field is written by chairperson.

```
<?xml version=”1.0”?>
<!DOCTYPE review_summary SYSTEM “review_target.dtd”>
<review_summary>
  <notification_date>12/31/01 0:0 AM</notification_date>
  <entry>
    <paper_title>Method for Parsing XML Document</paper_title>
    <paper_number>0120</paper_number>
    <author>Xerces</author>
    <review>
      <reviewer>Robert</reviewer>
      <rating>4.5</rating>
    </review>
    <result>Accept</result>
    <confirmation/>
  </entry>
  <entry>
    <paper_title>New Method for Stack Smashing Attack</paper_title>
    <paper_number>0123</paper_number>
    <author>Stackman</author>
    <review>
      <reviewer>Patrick</reviewer>
      <rating>4.0</rating>
    </review>
    <result>Accept</result>
    <confirmation/>
  </entry>
  <entry>
    <paper_title>Fantastic Public Key Cryptosystem</paper_title>
    <paper_number>0129</paper_number>
    <author>Dreamer</author>
    <review>
      <reviewer>Richard</reviewer>
      <rating>1.5</rating>
    </review>
    <result>Reject</result>
    <confirmation/>
  </entry>
</review_summary>
```

A set of access control policies is described as follows:

1. Authors submit their papers to the submission server. A chairperson assigns one or more reviewers to each submitted paper.
2. The reviewers can read an assigned paper except for the author’s name and evaluate it.
3. The program committee members can read reviewers’ evaluations and decide whether or not each paper should be accepted.
4. The chairperson can write each <result> element as a list of accepted papers.
5. The authors receive notifications of acceptance or rejection.

Primary Process Flow: Author (Xerces) makes a request of his submitted paper

Author Xerces submits the request: `Can Xerces read review document?`

PDP determines that Xerces can read his paper-related information but he cannot see the reviewer’s name of his paper and related information such as a rating score. Xerces’s view is as follows:

```
<?xml version=”1.0”?>
<!DOCTYPE review_summary SYSTEM “review_target.dtd”>
<review_summary>
  <entry>
    <paper_title>Method for Parsing XML Document</paper_title>
    <paper_number>0120</paper_number>
    <author>Xerces</author>
    <confirmation/>
  </entry>
</review_summary>
```

KeyPoints

- How to determine who is the author and to which portion s/he can make an access? There are two ways.
  - The first is to write author’s name in the summary document and PDP refers such information in the access evaluation time. (there is a <author> element in the review document). The access control policy has condition element that checks the requester’s id and the value of the author element.
  - The second is to create a system level attribute called owner or creator. The default policy such as an owner (or creator) can view could ease the writing of access control policy rules.

Glossary:

Access mode
Read, write, delete, and create

Access request
Data submitted by a requester that contains target XML resource, target node (element or attribute), access mode, and an authenticated user identity and its attributes.

Access response
Data returned by a PDP that contains access decision (grant or deny), target node, access mode, user information, and additional conditions (provisions, advise, etc.)

Attribute
Sub-structure of the target XML resource

Element
Sub-structure of the target XML resource

Target resource
The resource that consists of sub-structure such as element and attribute.

References:

[XACL1] XML Access Control Language (XACL) - http://www.trl.ibm.com/projects/xml/xacl/index.htm

[XACL2] XACL reference implementation - http://alphaworks.ibm.com/tech/xmlsecuritysuite

Appendix: XACL Policy Example

```
<policy xmlns=”http://www.trl.ibm.com/projects/xml/xacl”>
<!-- ==================================================
  1. The chairperson and the committee members can read
     the review document (unless a policy explicitly
     specifies denial.)
 =================================================== -->
  <xacl>
    <object href="/review_summary"/>
    <rule>
      <acl>
        <subject>
          <group>chair</group>
        </subject>
        <subject>
          <group>committee</group>
        </subject>
        <action name="read" permission="grant"/>
      </acl>
    </rule>
  </xacl>
<!-- ===================================================
  2. The chairperson can write the result (accept or
     reject) in the result field.
 =================================================== -->
  <xacl>
    <object href="/review_summary/entry/result"/>
    <rule>
      <acl>
        <subject>
          <group>chair</group>
        </subject>
        <action name="write" permission="grant"/>
      </acl>
    </rule>
<!-- ===================================================
   3. Authors cannot read the result of their own
      submission until the notification date comes.
 =================================================== -->
    <rule>
      <acl>
        <subject>
          <group>author</group>
        </subject>
        <action name="read" permission="deny"/>
        <condition operation="or">
          <predicate name="compareStr">
            <parameter value="neq"/>
            <parameter><function name="getValue">
              <parameter value="../author"/></function></parameter>
            <parameter><function name="getUid"/></parameter>
          </predicate>
          <predicate name="compareDate">
            <parameter value="before"/>
            <parameter><function name="getDate"/></parameter>
            <parameter>
              <function name="getValue">
                <parameter value="/review_summary/notification_date"/>
              </function>
            </parameter>
          </predicate>
        </condition>
      </acl>
    </rule>

<!-- ===================================================
   4. Authors can read the result of their own
      submission provided the read access is logged.
 =================================================== -->
    <rule>
      <acl>
        <subject>
          <group>author</group>
        </subject>
        <action name="read" permission="grant">
          <provisional_action name="log" timing="after"/>
        </action>
        <condition operation="and">
          <predicate name="compareStr">
```

```
            <parameter value=”eq”/>
            <parameter><function name=”getValue”>
              <parameter value=”../author”/></function></parameter>
            <parameter><function name=”getUid”/></parameter>
          </predicate>
          <predicate name=”compareDate”>
            <parameter value=”after”/>
            <parameter><function name=”getDate”/></parameter>
            <parameter>
              <function name=”getValue”>
                <parameter value=”/review_summary/notification_date”/>
                     </function>
            </parameter>
          </predicate>
        </condition>
      </acl>
    </rule>
  </xacl>
<!-- ====================================================
  5. Authors can read the their own submission entry.
 ==================================================== -->
  <xacl>
    <object href=”/review_summary/entry”/>
    <rule>
      <acl>
        <subject>
          <group>author</group>
        </subject>
        <action name=”read” permission=”grant”/>
        <condition operation=”and”>
          <predicate name=”compareStr”>
            <parameter value=”eq”/>
            <parameter><function name=”getValue”>
              <parameter value=”./author”/></function></parameter>
            <parameter><function name=”getUid”/></parameter>
          </predicate>
        </condition>
      </acl>
    </rule>
  </xacl>
<!-- ====================================================
  6. For anonymity purpose, the committee members and
     reviewers cannot read the authors’ names.
 ==================================================== -->
  <xacl>
    <object href=”/review_summary/entry/author”/>
    <rule>
      <acl>
        <subject>
          <group>committee</group>
        </subject>
        <subject>
          <group>reviewer</group>
        </subject>
        <action name=”read” permission=”deny”/>
      </acl>
    </rule>
  </xacl>
<!-- ====================================================
  7. For anonymity purpose, the committee members cannot
     read the reviewers’ names except for the case that
     the request initiator’s name is same as the
     reviewer’s name.
 ==================================================== -->
  <xacl>
    <object href=”/review_summary/entry/review/reviewer”/>
    <rule>
      <acl>
        <subject>
          <group>committee</group>
        </subject>
        <action name=”read” permission=”deny”/>
        <condition operation=”and”>
          <predicate name=”compareStr”>
            <parameter value=”neq”/>
            <parameter><function name=”getValue”>
```

```
              <parameter value=”.”/></function></parameter>
            <parameter><function name=”getUid”/></parameter>
          </predicate>
        </condition>
      </acl>
    </rule>
  </xacl>
<!-- ==================================================
  8. Authors cannot read reviewers’ evaluations.
 ================================================== -->
  <xacl>
    <object href=”/review_summary/entry/review”/>
    <rule>
      <acl>
        <subject>
          <group>author</group>
        </subject>
        <action name=”read” permission=”deny”/>
      </acl>
    </rule>
<!-- ==================================================
  9. Reviewers can read and write the review fields
     only for papers assigned to them.
 ================================================== -->
    <rule>
      <acl>
        <subject>
          <group>reviewer</group>
        </subject>
        <action name=”read” permission=”grant”/>
        <action name=”write” permission=”grant”/>
        <condition operation=”and”>
          <predicate name=”compareStr”>
            <parameter value=”eq”/>
            <parameter><function name=”getValue”>
              <parameter value=”./reviewer”/></function></parameter>
            <parameter><function name=”getUid”/></parameter>
          </predicate>
        </condition>
      </acl>
    </rule>
  </xacl>
<!-- ==================================================
  10. Reviewers can read titles and numbers of papers
     assigned to them.
 ================================================== -->
  <xacl>
    <object href=”/review_summary/entry/paper_title”/>
    <object href=”/review_summary/entry/paper_number”/>
    <rule>
      <acl>
        <subject>
          <group>reviewer</group>
        </subject>
        <action name=”read” permission=”grant”/>
        <condition operation=”and”>
          <predicate name=”compareStr”>
            <parameter value=”eq”/>
            <parameter><function name=”getValue”>
              <parameter value=”../review/reviewer”/></function></parameter>
            <parameter><function name=”getUid”/></parameter>
          </predicate>
        </condition>
      </acl>
    </rule>
  </xacl>
</policy>
```

Provisional Access Control on XML Resources

Title:	Provisional Access Control on XML Resources
Terse Description:	Online Contracting
Version:	v1.0
Submitted By:	Michiharu Kudo (IBM)
Date:	September 3, 2001

Summary

This use case presents another way of applying provisional actions such as verifying digital signature in online contracting application

Scope

Any resource represented in XML can be a target

Actors

Assumptions

- Target online contract document is written in XML. (XML with/without schema definition.)
- Access control policies are represented as a set of 4-tuple <object, subject, action, condition>.
- A client has a public-key pair for digital signature and send a signature value if required by the on-line contracting application system.

Non-technical Factors:

Process Sequence

Overview:

Consider that there is an electronic contract document represented in XML and there are two roles: business owner who offers business to clients; and the registered client who makes the contract with the business owner. Figure below illustrates the target XML contract document and the subject relation, which are stored in the authorization information repository. In the following examples, for brevity we use an element name for referring the target object.

```
<contractor level=”1”>
 <contract class=”A”>
  <t_and_c>Purchase of $1M over one year</t_and_c>
  <representative/>
 </contract>
 <comments>We accept the contract</comments>
</contractor>
```

Access control policy:

1. Business Owner can write t_and_c if t_and_c field is empty provided the access is logged

2. The Registered Client can write comments element if t_and_c element is not empty and comments element is empty, provided access is logged and the signature on the comments sent from the client is verified successfully
3. The Registered Client can read contractor subtree

Primary Process Flow: Read contract document

Access request: *Can Registered Client read the contract document?*

The PDP makes an access decision against the access request based on the access control policy rules. Decisions are the following:

- The Registered Client can read contractor element of the contract document.

The PDP makes an access decision against the access request based on the access control policy rules. Decisions are that the registered client can read all elements of the contract document.

Alpha Process Variant:

Write comments in the contract document

Access request: *Can Registered Client write comments element?*

The PDP makes an access decision against the access request based on the access control policy rules. Decisions are as follows:

- The Registered Client can write comments element if t_and_c element is not empty and comments element is empty, provided access is logged and the signature on the comments sent from the client is verified successfully

See Appendix A for checking the updated contractor document. The PEP also generates a log entry of this update access because the access decision asks to do so. See Appendix B.

Key Points

- PDP determines that the registered client who tries to write a comment must send his/her signature simultaneously and signature verification needs to be done before writing operation is executed. This scenario is a bit complicated and may be not so practical. However, the point is that provisional action is extensible to be capable of handling any operations like encryption, water-marking, and charging fees.
- XML can deal with security-related data structure as a first class object. It can be easily embedded in the source data structure or vise versa. This is one of the advantage of using XML as a target data structure.

Appendix A: XACL Sample Policy

```
<!------------------------------------------------------------------------------------------>
<!-- First xacl says that Business Owner can write t_and_c if t_and_c
 field is empty provided the access is logged-->
<xacl>
 <object href="/document/contractor/contract/t_and_c"/>
 <rule>
  <acl>
   <subject><roles><role>Business Owner</role></roles></subject>
   <action name="write" permission="grant">
    <provisional_action timing="before" name="log"/>
   </action>
   <condition operation="and">
    <predicate name="compareStr">
     <parameter>eq</parameter>
     <parameter><function name="get_field"/></parameter>
```

```
      <parameter>t_and_c</parameter>
      <parameter/>
     </predicate>
    </condition>
   </acl>
  </rule>
 </xacl>
 <!--------------------------------------------------------------------------------------->
 <!-- Second xacl says that the Registered Client can write comments element if t_and_c
 element is not empty and comments element is empty, provided access is logged and the
 signature on the comments sent from the client is verified successfully -->
 <xacl>
  <object href="/document/contractor/comments"/>
  <rule>
   <acl>
    <subject><roles><role>Registered Client</role></roles></subject>
    <action name="write" permission="grant">
    <provisional_action timing="before" name="log"/>
    <provisional_action timing="before" name="verify">
     <parameter>
      <SignedInfo>
       <CanonicalizationMethod Algorithm="http://www.w3.org/TR/1999/WD-xml-c14n-19991115"/>
       <SignatureMethod Algorithm="http://www.w3.org/2000/01/xmldsig/rsa-sha1"/>
       <Reference>
        <Transforms>
         <Transform Algorithm="http://www.w3.org/TR/1999/WD-xml-c14n-19991115"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/01/xmldsig/sha1"/>
       </Reference>
      </SignedInfo>
      </parameter>
     </provisional_action>
    </action>
    <condition operation="and">
     <predicate name="compareStr">
      <parameter>neq</parameter>
      <parameter><function name="get_field"/></parameter>
      <parameter>t_and_c</parameter>
      <parameter/>
     </predicate>
     <predicate name="compareStr">
      <parameter>eq</parameter>
      <parameter><function name="get_field"/></parameter>
      <parameter>comments</parameter>
      <parameter/>
   </predicate>
    </condition>
   </acl>
  </rule>
 </xacl>
 <!--------------------------------------------------------------------------------------->
 <!-- Third xacl says that the Registered Client can read contractor subtree. -->
 <xacl>
  <object href="/document/contractor"/>
  <rule>
   <acl>
    <subject><roles><role>Registered Client</role></roles></subject>
    <action name="read" permission="grant">
   </acl>
  </rule>
 </xacl>
 </policy>
```

Appendix B: XACL Sample Log Data

```
 <log href="/document/contractor/comments" time="5/10/00">
  <subject>
   <uid>CN=Satoshi Hada, OU=TRL, O=IBM, C=JP</uid>
   <roles><role>Registered Client</role></roles>
  </subject>
  <action permission="grant" name="write">
   <parameter>
    <Signature xmlns="http://www.w3.org/2000/01/xmldsig/">
     <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/1999/WD-xml-c14n-19991115"/>
      <SignatureMethod Algorithm=”http://www.w3.org/2000/01/xmldsig/rsa-sha1”/>
      <Reference IDREF=”Res0”>
        <Transforms>
         <Transform Algorithm=”http://www.w3.org/TR/1999/WD-xml-c14n-19991115”/>
        </Transforms>
        <DigestMethod Algorithm=”http://www.w3.org/2000/01/xmldsig/sha1”/>
        <DigestValue Encoding=”http://www.w3.org/2000/01/xmldsig/base64”>
                 WjDP4Pbe1KGFEHZHpPHI967w4SA=
        </DigestValue>
       </Reference>
      </SignedInfo>
      <SignatureValue>NJ0z/nrH0MXy5XQW…</SignatureValue>
      <KeyInfo>
       <X509Data>
 <509Name>CN=Satoshi Hada, OU=TRL, O=IBM, C=JP</X509Name>
       <X509Certificate>MIIB7TCCAVYCBD1…</X509Certificate>
       </X509Data>
      </KeyInfo>
      <dsig:Object Id=”Res0” xmlns=”” xmlns:dsig=”http://www.w3.org/2000/01/xmldsig/”>
       <parameter>We accept the contract</parameter>
      </dsig:Object>
     </Signature>
    </parameter>
   <provisional_action timing=”before” name=”log”/>
   <provisional_action timing=”before” name=”verify”>
    <parameter>
     <SignedInfo>
      <CanonicalizationMethod Algorithm=”http://www.w3.org/TR/1999/WD-xml-c14n-19991115”/>
      <SignatureMethod Algorithm=”http://www.w3.org/2000/01/xmldsig/rsa-sha1”/>
      <Reference>
       <Transforms>
        <Transform Algorithm=”http://www.w3.org/TR/1999/WD-xml-c14n-19991115”/>
       </Transforms>
       <DigestMethod Algorithm=”http://www.w3.org/2000/01/xmldsig/sha1”/>
      </Reference>
     </SignedInfo>
    </parameter>
   </provisional_action>
  </action>
 </log>
```

Static Access Control on XML Resources

Title:	Static Access Control on XML Resources
Terse Description:	Configuration file management
Version:	v1.0
Submitted By:	Michiharu Kudo (IBM)
Date:	September 3, 2001

Summary

This use case presents four scenarios for retrieving and updating a sort of configuration files represented in XML in fine-grained manner. Access decision is determined statically.

Scope

Any resource represented in XML can be a target

Actors

Assumptions

- Target configuration file is written in XML. (XML with/without schema definition.)
- Access control policies are represented as a set of triplet <object, subject, action>.
- Some users are not allowed to read/update specific element and/or attribute within the target resource.
- XACML provides necessary and useful set of schema definition for subject, object, action, and additional information. For example, four kinds of actions such as read, write, create, and delete are reasonable set for browsing, and updating XML document here. At the same time, they should be extensible in accordance with future requirements.
- Access control policies are defined without any environmental and run-time values.

Non-technical Factors:

Process Sequence

Primary Process Flow: Read access control scenario

It is often the case that some elements and/or attributes of the configuration contents are read only by a specific user (e.g. Website maintainer.) First a requester sends an access request that consists of a target XML configuration file using URI and an access mode such as “http://appl.server/config.xml” and “read”, respectively. The application server (PEP) retrieves “config.xml” and calls the PDP with the root node pointer, server-authenticated user id and his/her attributes, a “read” action, and optionally the contents of “config.xml”. A sample “config.xml”is as follows:

```
<?xml version=”1.0”?>
<configuration>
  <docRoot type=”default”>/</docRoot>
  <passwd_hints type=”MaidenName”>Alice</passwd_hints>
  <qos_policy type=”normal”>qos.xml</qos_policy>
</configuration>
```

Access control policy:

1. Administrator can read root node (<configuration> element) and all the descendant nodes.
2. Website maintainer can read only <docRoot> element and its descendant nodes.

Access request: *Can Website maintainer read the root node?*

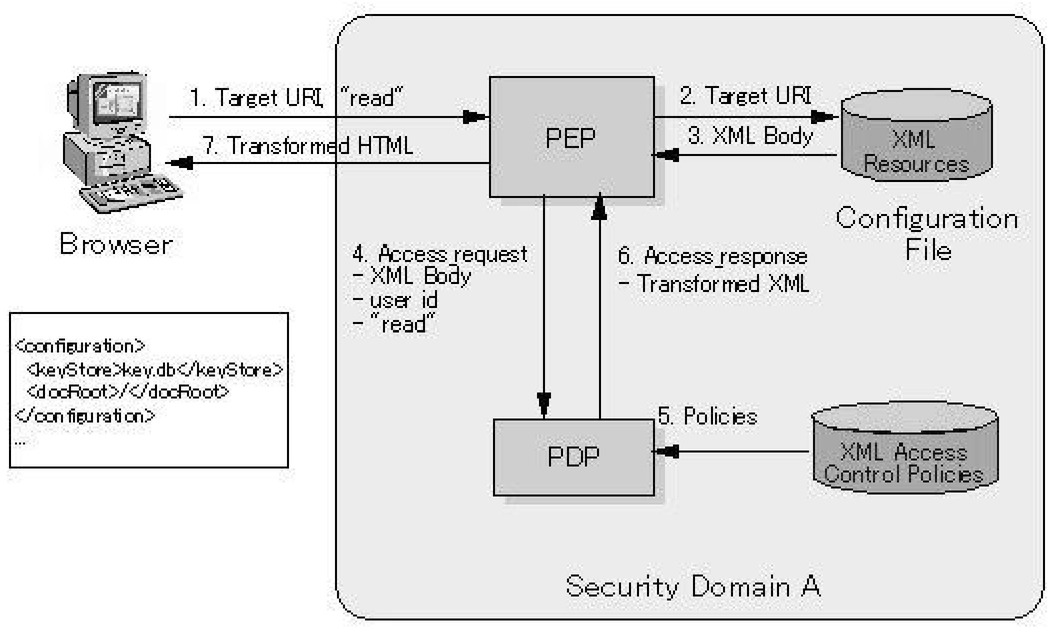
The PDP makes an access decision against the access request based on the access control policy rules. Decisions are the following:

- Website maintainer can read <docRoot> element and its descendant nodes but cannot read <passwd_hints> or <qos_policy> elements and their descendant nodes.

The PDP (or PEP) transforms original “config.xml” to data for requester described below. The transformed data only includes permitted nodes. The PEP sends the transformed “config.xml” back to the requester.

```
<?xml version=”1.0”?>
<configuration>
  <docRoot type=”default”>/</docRoot>
</configuration>
```

Flow Diagram:



```
<configuration>
  <keyStore>key.db</keyStore>
  <docRoot>/</docRoot>
</configuration>
...
```

Key Points

- Sub-structure (node such as element and attribute) is referred using Xpath (or similar technique) both in the access control request and the access control policy.
- An access request must contain a node reference pointer that points a specific node. It may implicitly mean the access request to all the descendant nodes below the explicitly specified target node.
- One access response may contain a set of access decisions on each descendant nodes of the requested target node. The access response may include additional information such as provisions and advise.

- Decision-making process in PDP may include several meta-policies such as default policy, propagation policy, and conflict resolution policy.
    - Default policy (grant or deny) is applied when there are no explicit access control rules for the target node in the policy statement.
    - Propagation policy specifies how one access decision can derive to the descendant nodes (or ancestor nodes).
    - Conflict resolution policy determines access decision if conflict occurs (both grant and denial decision are derived on the same node)
- Schema definition may be checked before read operation is enforced.
- Target XML resource may not have a schema definition, may have DTD definition, XML Schema definition, or other schema definitions. The system must allow this flexibitliies.

Alpha Process Variant: Write access control scenario

This is similar to the previous scenario except for the access mode.

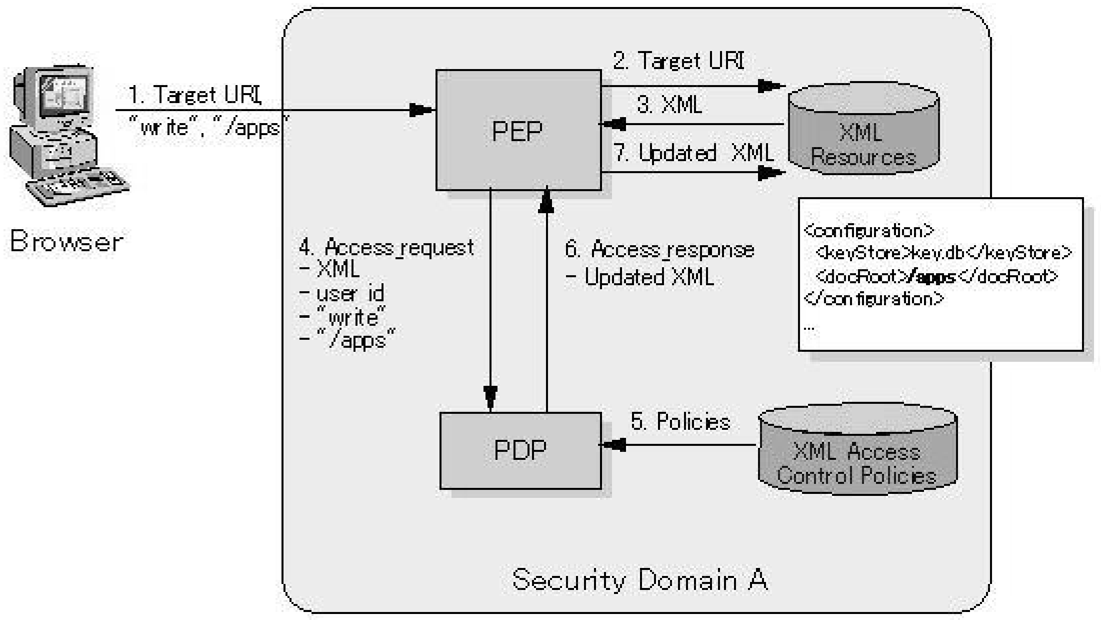
First a requester sends an access request that consists of a target URI such as “http://appl.server/config.xml”, element or attribute pointer, an access mode “write”, and its argument. The PEP retrieves “config.xml” and calls the PDP with the target reference node, requester’s user id, a “write” access mode, the argument to be written, and optionally the contents of “config.xml”. The PDP makes an access decision based on the access control policy rules and update the target node of “config.xml” with the given argument. The PEP (or PDP) changes the original “config.xml” if the access decision is positive.

Access control policy:

1. Administrator can write the root node, provided the access is logged.
2. Website maintainer cannot write any node.
3. Access request: *Can Administrator update the <qos_policy> element with “qos2.xml”?*
4. The PDP makes an access decision against the access request based on the access control policy rules. Decision is the following:
5. Administrator can update <qos_policy> element with “qos2.xml”, provided the access is logged.
6. The PEP updates the original document and stores the access decision to the log file.

```
<?xml version=”1.0”?>
<configuration>
  <docRoot type=”default”>/</docRoot>
  <passwdHints type=”MaidenName”>Alice</passwdHints>
  <qos_policy type=”normal”>qos2.xml</qos_policy>
</configuration>
```

Flow Diagram:



Key Points

- An access request may include a set of argument such as a value to be written in.
- Access control policies may include parameters (e.g. regular expressions) such as the value to be written in. (e.g. Website maintainer can update the <docRoot> element with "/htdocs/*")
- The access response may include additional information such as a set of provisions and advices.

Beta Process Variant: Delete access control

This is similar to the previous scenario except for the access mode.

First a requester sends an access request that consists of a target URI such as http://appl.server/config.xml, an element or attribute pointer (/configuration/qos_policy), and an access mode "delete". The PEP retrieves "config.xml" and calls the PDP with target node pointer, requester's user id, a reference pointer, a "delete" access mode, and optionally the contents of "config.xml". The PDP makes an access decision based on the access control policy. The PEP (or PDP) deletes the target node of the original "config.xml" if the access decision is positive.

Access control policy:

1. Administrator can delete the <qos_policy> element, provided the access is logged.
2. Access request: *Can Administrator delete the <qos_policy> element?*
3. The PDP makes an access decision against the access request based on the access control policy rules. Decision is the following:
4. Administrator can delete <qos_policy> element and all the descendant nodes, provided the access is logged.

The resultant target document becomes as follows:

```
<?xml version="1.0"?>
<configuration>
```

```
  <docRoot type=”default”>/</docRoot>
  <passwdHints type=”MaidenName”>Alice</passwdHints>
</configuration>
```

Flow Diagram:

- The same with the alpha process variant.

Key Points:

- A delete operation means the deletion of the target node as well as all the descendant nodes.
- In case of delete operation, different propagation policy may be needed. For example, usual policy may allow deletion only when all the descendent nodes are evaluated as deletable.

Gamma Process Variant: Create access control

This is similar to the previous scenario except for the access mode.

First a requester sends an access request that consists of a target URI such as http://appl.server/config.xml, an element or attribute pointer (e.g. /configuration), an access mode “create” with an argument e.g. <fw_policy>firewall.xml</fw_policy> The PEP retrieves “config.xml” and calls the PDP with the target pointer, requester’s user id, a “create” access mode, the argument to be created, and optionally the contents of “config.xml”. The PDP makes an access decision based on the access control policy rules. The PEP (or PDP) creates new nodes if the access decision is positive. Resultant target resource becomes:

```
<?xml version=”1.0”?>
<configuration>
  <docRoot type=”default”>/</docRoot>
  <passwdHints type=”MaidenName”>Alice</passwdHints>
  <fw_policy>firewall.xml</fw_policy>
</configuration>
```

Flow Diagram

- The same with the alpha process variant.

Key Points

- In case of creation operation, more parameters may be needed e.g. the position of the creation of the new node.
- Schema definition may be checked before creation operation is executed.

Glossary

Access mode
Read, write, delete, and create

Access request
Data submitted by a requester that contains target XML resource, target node (element or attribute), access mode, and an authenticated user identity and its attributes.

Access response
Data returned by a PDP that contains access decision (grant or deny), target node, access mode, user information, and additional conditions (provisions, advise, etc.)

Attribute
Sub-structure of the target XML resource

Element
Sub-structure of the target XML resource

Target resource
The resource that consists of sub-structure such as element and attribute.

References

[XACL1] XML Access Control Language (XACL): - http://www.trl.ibm.com/projects/xml/xacl/index.htm

[XACL2] XACL reference implementation: http://alphaworks.ibm.com/tech/xmlsecuritysuite

Security Policies for Workflow

Title:	Security Policies for Workflow
Terse Description:	Workflow is a multi-step electronic transaction in which several security policies may participate at each stage.
Version:	1.0
Submitted By:	Carlisle Adams
Date:	September 5, 2001

Summary

In an e-business environment, some transactions may take place over multiple steps, involving several processes, several platforms, and one or more interactions with human entities (although typically the goal is to automate as much of the transaction as possible). Each step in the transaction may be envisioned as one or more input values (data, request, etc.), a data processing or data transformation stage, and an output result sent to one or more next steps. Each of these three sub-steps may be governed by an appropriate security policy. XACML may encompass all such relevant security policies in its language, but at the very least needs to acknowledge their potential existence in its model.

Scope

Actors

Assumptions

Non-technical Factors

Process Sequence

Primary Process Flow

1. Input data at a step in the transaction needs to be of the “proper” form, with respect to security operations. [Has it been signed by the appropriate entities or roles? Has it been encrypted for the appropriate entities or roles? Has it been time-stamped? Is it accompanied by a receipt from an archive service? Is the sender authenticated and authorized to have sent this data? Are the required SAML assertions or other relevant data available?] This is analogous to checking an input XML document against a schema for that document type, but the focus is on the security properties of the document, rather than merely its syntax.
2. The data needs to be processed or transformed in some way. [Does it (all of it, or selected elements) need to be signed or encrypted? By whom or for whom? Does it need to be time-stamped or archived? Do SAML assertions or artifacts need to be generated and sent with the output data? How does the data need to be transformed (e.g., is any filtering of the elements necessary)? Are there conditions or conditional actions that need to be checked or performed?] This is similar to some of the access control rules already under consideration in other use cases, but is somewhat richer and more general because it needs to embrace a greater set of security operations and needs to account for the fact that the “requester” and the “recipient” are different entities (with their own policies).

3. The resulting data needs to be sent to the next step(s) in the transaction. [Does the potential receiver need to be authenticated? Does the receiver's authorization to receive the data need to be checked? Has the requester stipulated any constraints on the audience or use of this data, and does this match the potential receiver and the uses to which it claims received data will be put? If any of the conditions or conditional actions fail, does this process need to be "rolled back" to some previous step in the transaction (how far, and how is this done)?] Again, this is similar to some of the relevant considerations in other use cases, but is richer and more general.

Flow Diagram

Key Points

- The main point is that several security policies may participate at each step in the workflow transaction
    - requester policy describing who can receive the data and what they can do with it
    - receiver policy describing what data sent to them should look like and what they intend to do with it
    - policy describing "proper" input data (with respect to security properties)
    - policy describing the required security processing or transformations of data at that step
    - policy describing what checks need to be made prior to sending data out (including conditional actions and roll-back procedures).
- XACML may consider developing a language rich enough to support all these types of policies, or may decide that some of this work is being done elsewhere (e.g., W3C P3P). In any case, however, XACML needs to be aware of these concepts and acknowledge them in its overall policy model.

Glossary

References

Microsoft.NET Stack Walk

Title: Microsoft.NET Stack Walk Terse

Description: Evaluation of permissions and resulting execution of code in a Microsoft.NET environment.
Version: 1.0

Submitted By: Carlisle Adams (but entirely derived from the circulated Microsoft document)

Date: September 7, 2001

Summary

The permissions associated with each point in the entire calling sequence are evaluated before the actual requester is granted or denied access.

Scope

Actors

Assumptions

Non-technical Factors

Process Sequence

Primary Process Flow

1. At run time, permissions are evaluated based on the execution of code.
2. An assembly, "A3," provides its evidence, along with evidence from the host, to the policy evaluator.
3. The evaluator also takes the permission requests from the assembly into consideration in creating a grant of permissions, "G3."
4. Assembly A3 is called by assembly A2, which has been called by assembly A1.
5. When assembly A3 performs an operation that triggers a security check, the permission grants of A2 and A1 are also examined to ensure that they have the permissions requested by A3.

In this process, which is called stack walking, the permission grants of every assembly on the stack are inspected to see whether the grant set contains the permission demanded by the security check. If each assembly on the stack has been granted the permission demanded by the security check, the call succeeds. If any assembly has not been granted the demanded permission, the stack walk will fail, and a security exception will be thrown. The code-access security stack walk protects code against the "luring" attack. In this common attack, malicious code tricks more trusted code into doing something it can't do alone-effectively leveraging the good code's permissions for ill intent. This kind of attack is extremely difficult for developers to guard against, but the stack walk ensures that if lower-trust code is involved, the permissions available are reduced to that of the lowest-trusted code. The result is that code may be acquired from sources with varying degrees of trust, and run with restrictions appropriate to the particular context of that code's execution.

Flow Diagram

Key Points

- The interesting thing about this use case is that it highlights a requirement similar to delegation, but not identical.
    - The permissions of every piece of code in the calling sequence are checked, but these permissions may (and likely will) be assigned independently (perhaps by different authorities, particularly for code downloaded from the Web).
    - When one routine calls another, it is not delegating any privileges to that called routine; the called routine must have its own set of privileges.
- In some ways, this requirement is similar to the concept of co-signing for documents.
    - As with delegation, the privileges of several entities are checked (not just the immediate requester, but others as well in a kind of chain), but no delegation of privilege has occurred from one of these entities to another.

Glossary

References

Document from Microsoft to SAML f2f meeting: August 2001

Provision User for Third Party Service

Title:	Provision User for Third Party Service
Terse Description:	Create an account, profiles, and policies on behalf of a user in a managed, third party service.
Version:	v0.1
Submitted by:	Gilbert W. Pilz Jr.
Date:	September 7, 2001

Summary

Service Aggregators provide a central locus through which users can subscribe to and access an array of individual services. These services may be provided within the Service Aggregators organization or by independent, third party organizations. This use case describes the process whereby a user is provisioned for a service.

Scope

User account/profile creation, authorization attributes queries, authorization policy creation.

Actors

- Administrative User
- Third Party Service PIP Entity
- Third Party Service Attribute Authority
- Third Party Service PDP Entity
- Aggregation Entity
- User

Assumptions

- The Service Aggregator and the Third Party Service Provider have a business relationship. Portions of this business relationship are reflected in the establishment of a trust relationship (implemented by the "appropriate" security protocols, exchange of keys and/or certificates, etc.) between the Service Aggregator and the Third Party Service. This trust relationship enables the Administrative User (hereafter called Admin) to create objects and manage policies with the Third Party Service. Establishment and maintenance of this trust relationship is out of the scope of this use case.
- Prior to this use case a "company" container object was created for this user. This container object serves to group and scope the individual users for the purposes of billing, auditing, and authorization. The creation of this company container is covered under the "Subscribe User Organization to Third Party Service" use case.
- Also implicit in this use case is the notion that the User has an existing account with the Aggregator. The account being created in this use case refers to the sub-account within the Third Party Service and not the primary account with the Aggregator.

Non-technical Factors

Process Sequence

Primary Process Flow

A user requests to be provisioned for a particular service. This request may be accompanied by modifiers such as “I need to be added to the ‘Auditors’ group for this service”. After reviewing and verifying the request, the Admin, acting through the Aggregation Entity, provisions the user for the requested service. This process may involve several steps. The ordering of these steps may vary between services but in general they involve.

1. The Admin creates an account for the user. This account should exist within the context of the previously created company container.
2. The Admin queries the Third Party Service PIP, PDP, and Attribute Authorities for the list of attributes relevant to the new user account.
3. The Admin assigns the proper authorization attributes to the newly created account. These authorization attributes are bounded by those permitted to be assigned by the aggregator, those permitted to users within the company container, and those requested by the originating user.
4. The Admin creates and assigns the authorization policy that controls who is allowed to modify or delete the newly created account.
5. The Aggregating Entity records the details (such as user name, designated name, etc.) of this account for later use during service access, billing, and auditing.

Flow Diagram

Key Points:

- Scalability and life cycle considerations prevent the Aggregation Entity from recording the authorization attributes (groups, roles, etc.) used by all the Third Party Services that it aggregates. Therefore it must be possible to query, at run time, for the all the authorization attributes that may be assigned to a particular user account.
- It is not necessary for an “authorization attribute query” to return **all** the attributes that may be relevant to any authorization decision (since this may be unknowable), simply those static attributes which may be assigned to a particular user account.
- In the absence of an explicit request to modify the authorization attributes of the newly created account, sensible default attributes should be assigned and/or inherited.
- In the absence of an explicit request to create an authorization policy that controls the newly created account, a sensible default policy should be assigned and/or inherited.

Alpha Process Variant: User Self-Provisioning

Under certain circumstances where it is deemed acceptable by the Aggregator and the Third Party Service Provider, users may be allowed to provision services for themselves. This variant functions identically to the primary process, the only difference being that the initiating actor is now the User and not the Admin.

Flow Diagram

Key Points:

- It should be possible to scope “authorization attribute query” by the authority to actually set that returned attributes on a given user account. For instance if, as a user, I am not allowed to add myself to the “Account Admins” group, then the authorization attribute query should not return this group as an attribute.

- Whether or not the authorization attribute queries returns only settable attributes, the attributes that the user is allowed to add to their own account should be limited by authorization policy.

Beta Process Variant: Automatic Provisioning

Some organizations that subscribe to the Service Aggregator may request that certain services be automatically provisioned for each new user that is created on behalf of their organization. This variant directly contradicts one of the stated assumptions; that the user already has an account with the Aggregator. Leaving aside the details on how the user account is created with the Aggregator, suffice it to say that this variant is invoked automatically by the Aggregator Entity during this process.

Flow Diagram

Key Points

- Because the primary actor is a programmatic entity, there is no way to interactively select the attributes to be assigned to the account created with the Third Party Service. Therefore the newly created account must be assigned or inherit default attributes and the policy used to control the account should be a default or an inherited policy.

Glossary

Service Aggregator
An organization that aggregates services provided by one or more Service Providers. Typical services offered by an Aggregator include single sign on, provisioning, billing, service monitoring and support.

Aggregating Entity
A software platform that carries out the functions of a Service Aggregator.

References

Subscribe User Organization to Third Party Service

Title:	Subscribe User Organization to Third Party Service
Terse Description:	Create a company object along with relevant policies and attributes to act as a container for users of a service.
Version:	V 0.1
Submitted By:	Gilbert W. Pilz Jr.
Date:	September 7, 2001

Summary

Service Aggregators provide a central locus through which users can subscribe to and access an array of individual services. These services may be provided within the Service Aggregators organization or by independent, third party organizations. Most service providers have some notion of a User Organization or Company which serves as a container object for all the Users that belong to that organization or company. This use case describes the process whereby an organization (company) is related to a Third Party Service Provider in such a way that members of the organization (users) can be provisioned with the provided service. In particular it outlines the creation/submission of new authorization policies that apply to the User Organization along with the creation/submission of new authorization attributes that may apply to members of the User Organization.

Scope

Company account/container creation, authorization attribute creation, authorization policy creation.

Actors:

- Administrative User
- User Organization
- Third Party Service Registry
- Third Party Service Attribute Authority
- Third Party Service PDP Entity
- Aggregation Entity

Assumptions

- The Service Aggregator and the Third Party Service Provider have a business relationship. Portions of this business relationship are reflected in the establishment of a trust relationship (implemented by the "appropriate" security protocols, exchange of keys and/or certificates, etc.) between the Service Aggregator and the Third Party Service. This trust relationship enables the Administrative User (hereafter called Admin) to create objects and manage policies with the Third Party Service. Establishment and maintenance of this trust relationship is out of the scope of this use case.
- Also implicit in this use case is the notion that the User Organization has an existing relationship with the Service Aggregator.

Non-Technical Factors

Process Sequence

Primary Process

1. A User Organization submits a request to subscribe to one of the services provided by the Service Aggregator.
2. After reviewing and approving the request the Admin interacts with Aggregation Entity to subscribe the User Organization to the service.

Portions of this process, such as recording the fact that Users belonging to the User Organization can be provisioned for the service, are internal to the Aggregation Entity itself and out of the scope of this use case. Other portions of this process, such as the creation of the User Organization Object, are carried out between the Aggregation Entity and various entities belonging to the Service Provider. These are also out of the scope of this use case. What is in the scope of this use case is the creation of one or more authorization policies that govern the use of the User Organization Object within the Service Provider along with the creation of one or more Authorization Attributes that are used by these and other policies.

For example suppose a new User Organization Object corresponding to a company called "Mavericks" is created within the "Acme" service. In addition to this object, a set of authorization policies are created that say (in effect) "only super user, aggregating admins, and members of 'Mavericks admins' are allowed to add users to this object; only super user, aggregating admins, and members of 'Mavericks users' are allowed to list the contents of this object, etc.". A number of these policies may call for the creation of new attributes such as "Mavericks admins" and "Mavericks users". On top of all this may also be the creation of a set of default policies such as "all Users created within the Mavericks object are automatically assigned the 'Mavericks users' attribute". These policies and attributes may be created automatically by the Service Provider or they may be created manually by the Aggregating Entity (acting on behalf of the Admin).

Flow Diagram

Key Points

- The creation of new User Organization Objects within a Service Provider are accompanied by the creation of new authorization policies for that object.
- These authorization policies may, in turn, require the creation of new attributes.
- Additional, default policies may also be created.

Glossary

Service Aggregator
An organization that aggregates services provided by one or more Service Providers. Typical services offered by an Aggregator include single sign on, provisioning, billing, service monitoring and support.

Aggregating Entity
A software platform that carries out the functions of a Service Aggregator.

User Organization Object
An object within a Service Provider system that acts as a container for Users within the system. This concept is synonymous with the idea of a "Company" business object.

References

Policy Analysis Use Case

Tile:	Policy Analysis Use Case
Terse Description:	Policies from multiple systems are aggregated in an independent system for analysis and reporting
Version:	1.0
Submitted by:	Ken Yagen
Date:	September 7, 2001

Summary

Access control policies from stove-piped systems as well as multiple access management systems that may exist inside an enterprise must be brought together an analyzed for policy consistency and adherence to corporate and industry rules and regulations.

Scope

Web and legacy applications, Custom business applications, packaged applications, third party access management systems, operating system and network security policies

Actors

- Originating System: Applications or systems maintaining the access control policies
- Policy Aggregator: System responsible for aggregating and analyzing the access control policies.
- Administrator: User of the Policy Aggregator system

Assumptions

- Access control policies and policy models can be extracted from the Originating Systems

Non-technical Factors

In a B2B scenario, not all systems may be contained in the same enterprise and there may be issues extracting policy from those systems.

Process Sequence

Primary Process Flow:

1. Policy and Policy Model are extracted from the Originating System
2. Policy and Policy Model are transformed into a format that is understood by the Policy Aggregator
3. Policy and Policy Model are fed into the Policy Aggregator
4. Administrator analyzes the policies contained in the Policy Aggregator

Key Points

- Each Originating System may employ different policy models
- Policy elements could be represented differently in each system and naming schemes could be arbitrary

Use Case Commonalities

Tile:	Use Case Commonalities
Terse Description:	Summary of commonalities across Draft Use Cases
Version:	1.0
Submitted by:	Ken Yagen
Date:	September 7, 2001

Use Case	Commonality	Description
Clinical Record (Record Addendum)	Admin Policy	Policy about who can make changes to policy
Online Access Control	Attribute Protocol	Policy input request/response between PDP and PIP
SAML Authz Decision	Attribute Protocol	Policy input request/response between PDP and PIP
Online Access Control	Attributes	Syntax for Policy Inputs
SAML Authz Decision	Attributes	Syntax for Policy Inputs
Online Access Control	Audit	Audit trail syntax for information recorded by PDP
Clinical Record (Break Glass)	Audit	Audit trail important
Online Access Control (Beta)	Conditions	Syntax when requesting additional information by PDP or requiring pre-conditions to be checked first before authorizing
Clinical Record (Patient Override)	Conditions	Rules regarding access can rely on outside factors
Online Contracting (Variant 2)	Conditions	Rules have pre conditions that must be applied before authorization allowed.
Online Contracting (Variant 2)	Conditions	Rules have post conditions that must be applied once authorization allowed
ebXML Registry	Location of Policy	Policy exists separate from objects
Clinical Record	Location of Policy	Syntax for access policy that can be embedded in the document
Clinical Record (Record Transmittal)	Location of Policy	Policy must travel with the document, even as it changes
Online Contracting (Variant 2)	Location of Policy	Policy must be able to be embedded in source document
Clinical Record (Patient Override)	Modificaiton	New access privileges may need to be applied
Clinical Record	Modification	Policy as written in document cannot change without resigning the document
Clinical Record (Record Addendum)	Modification	Policy and document can be addended.
Clinical Record (Record Retrieval)	Multiple Policies	Multiple policies may exist in document
Clinical Record (Break Glass)	Multiple Policies	Ability to have “superuser” rights to override policy in medical emergency
Groupware application	Multiple Policies	Users can have different roles at different times
Static access control	Multiple Policies	Multiple policies could apply; conflict resolution needed

Workflow	Multiple Policies	Policies may originate from different locations
ebXML Registry	Policy Language	Syntax represent policy submitted to registry with RegistryObject
ebXML Registry	Policy Language	limited number of privileges
ebXML Registry	Policy Language	UUID used to represent unique objects
ebXML Registry	Policy Language	Policy can be on user identity, role, or group
Clinical Record	Policy Language	Policy can be on individuals
Clinical Record	Policy Language	Objects in policy go to the document element level of granularity
Online Contracting	Policy Language	Access control on document elements
Online Contracting (Alpha Variant)	Policy Language	Rules have conditions based on transaction information
Groupware application	Policy Language	Limited number of privileges and roles
Groupware application	Policy Language	Element level control in document
Static access control	Policy Language	Schema for policy elements must be extensible
Static access control	Policy Language	Element level control in document
Static access control	Policy Language	Grant and deny rules
Online Catalog	Policy Language	Limited number of user groups
Online Catalog	Policy Language	Grant and deny rules
Online Catalog	Policy Language	Attribute level control on document
Policy Provisioning	Policy Protocol	Policy request/response between PRP and PDP
Policy Provisioning (Alpha)	Policy Protocol	Notification of policy updates available by PRP to PDP
ebXML Registry	Policy Protocol	Protocol for submitting policy to registry
Online Access Control	SAML Ext	Authorization request and decision syntax
SAML Authz Decision	SAML Ext	Authorization request and decision syntax