

TELECOMMUNICATION  
STANDARDIZATION SECTOR

TD 2365

STUDY PERIOD 2005-2008

English only

Original: English

Question(s): 9/17

Jeju, Korea, 19-28 April 2006

## TEMPORARY DOCUMENT

Source: Q.9 Rapporteur

Title: Draft of Guideline on Single Sign-On and Access Control Methods for Mobile Web Environments

**All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU**

This TD is a follow-up of the WP2 plenary meeting.

## Summary

As a security feature, Single Sign-On allows a user to log into many different services offered by the distributed systems while the user needs to authenticate identification only once and always in

<b>Contact:</b>	Heung Youl Youm SoonChunHyang University Korea	Tel: +82 41 530 1328 Fax: +82 41 530 1494 Email: <a href="mailto:hyyoum@sch.ac.kr">hyyoum@sch.ac.kr</a>
<b>Contact:</b>	Jae Seung Lee ETRI Korea	Tel: +82 42 860 1326 Fax: +82 42 860 5611 Email: <a href="mailto:jasonlee@etri.re.kr">jasonlee@etri.re.kr</a>
<b>Contact:</b>	Ki Young Moon ETRI Korea	Tel: +82 42 860 6644 Fax: +82 42 860 5611 Email: <a href="mailto:kymoon@etri.re.kr">kymoon@etri.re.kr</a>
<b>Contact:</b>	Kyo-Il Chung ETRI Korea	Tel: +82 42 860 1920 Fax: +82 42 860 5611 Email: <a href="mailto:kyoil@etri.re.kr">kyoil@etri.re.kr</a>
<b>Contact:</b>	Dongkyoo Shin Sejong University Korea	Tel: +82 2 3408 3242 Fax: +82 3030 498 4273 Email: <a href="mailto:shindk@sejong.ac.kr">shindk@sejong.ac.kr</a>
<b>Contact:</b>	Jongil Jeong Sejong University Korea	Tel: +82 2 498 4273 Fax: +82 3030 498 4273 Email: <a href="mailto:jjjeong@gce.sejong.ac.kr">jjjeong@gce.sejong.ac.kr</a>

**Attention:** This is not a publication made available to the public, but an **internal ITU-T Document** intended only for use by the Member States of ITU, by ITU-T Sector Members and Associates, and their respective staff and collaborators in their ITU related work. It shall not be made available to, and used by, any other persons or entities without the prior written consent of ITU-T.

the same way. Therefore, Single Sign-On is very suitable for the distributed mobile environments. Thus, Single Sign-On should be considered for constructing secure and extensible mobile Web Services environments. For secure mobile Web Services, access control to the resources in the service domain also has to be provided with the Single Sign-On architecture. Currently Web Services offer the schema for Policy element for access control as a consideration for security, and show the structure for tModel, which includes Policy element. However practical guideline is not provided for implementing access control mechanism for it.

This draft of 'Guideline on Single Sign-On and Access Control Methods for Mobile Web Environments' describes Single Sign-On and access control methods for mobile web environments using SAML and XACML. This draft also describes access control methods for UDDI in web services using XACML. One of the purposes of this draft is to provide good usage models of SAML and XACML that have been standardized in ITU-T SG17.

## 1. Scope

The scope of this contribution deals with the Single Sign-On and access control methods for mobile web environments based on SAML and XACML. Access control methods for UDDI in web services using XACML are also covered.

## 2. References

The following ITU-T Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

*(To Be Developed)*

- ITU-T Recommendation X.websec-1, *SAML Version 2.0*, (2006)
- ITU-T Recommendation X.websec-2, *XACML Version 2.0*, (2006)

## 3. Definitions

*(To Be Developed)*

## 4. Abbreviations

*(To Be Developed)*

<b>PDP</b>	Policy decision point
<b>PEP</b>	Policy enforcement point
<b>SAML</b>	Security Assertion Markup Language
<b>SSO</b>	Single Sign-On
<b>XML</b>	eXtensible Markup Language

## 5. Overview

In the scope of Web Services implementation, following security factors should be considered in order to ensure the secure Web Services environments.

- Identification
- Authentication
- Authorization
- Integrity
- Confidentiality
- Non-Repudiation

**Table 1 – the comparison of authentication method**

Methods Factors	PKI	Username/password	Kerberos
Identification	O	O	O
Authentication	O	O	O
Authorization	O	X	O
Integrity	O	X	O
Confidentiality	O	O	O
Non-Repudiation	O	X	O

Table 1 shows the representative authentication methods widely adapted in most Web-based applications. As shown in Table 1, key-based methods (PKI, Kerberos) are satisfying all security factors while non key-based method (username/password) is satisfying some security factors.

**Table 2 – Features of authentication methods in distributed environments**

Methods Features	PKI	Username/password	Kerberos
Repetition of security information offer	X	O	X
Frequent transmission of passphrase	X	O	X
Mutual authentication	O	X	O
Exchange of private key	X	X	O
Necessity of additional infrastructure	O	X	O
Level of OSI 7 Layer	4th	4th	7th
Necessity of encryption for transport layer	O	O	X
Authentication scope limited to single domain	X	O	O
Support of access control	O	X	O

Table 2 shows the comparison of when authentication scheme is established using only one of the presented authentication methods in distributed environments. As shown in Table 2, although end-to-end security such as PKI and Kerberos need not offer security information repeatedly, it needs additional infrastructure. In contrast with end-to-end security, although point-to-point security such as Username/password does not need additional infrastructure, it requires users to offer security information repeatedly. Although only key-based methods meet all the security factors, it is recommended to establish the authentication scheme of Web Services combining key-based and non key-based method because Web Services can be constructed by combining various and different domains. For example, in case of combination of key-based environment and non key-based environment, a user, who belongs to an environment in which certificates are not supported, is allowed to log into non key-based environments using username/password and then the user can access to key-based and ad hoc environment using certificate or Kerberos ticket.

The example mentioned above describes the concept of Single Sign-On (SSO). It is the basis of the implementation of SSO to share authentication information generated from a domain with other trusted domains and exchange it. This basis allows a user to log into many different services offered by the distributed systems while the user needs to authenticate identification only once and always in the same way. Thus, SSO is very suitable and necessary authentication scheme for the distributed mobile Web Services environments.

**Table 3 – Comparison of Single Sign-On implementation methods**

Approach	Method	Kind of security token
Central repository	Passport	
	Liberty Alliance	
Message attachment	Kerberos	Kerberos ticket
	WS-Security	X.509 certificate, Kerberos ticket, username/password
	SAML	Kerberos ticket, Password, Name Identifier (application defined), various of keys, Secure Remote Password (SRP), Hardware token, SSL/TLS Certificate Based Client Authentication, X.509 public key, PGP public key, SPIK Public key, XML Digital Signature, Unspecified

Table 3 shows two representative approaches for implementing SSO. The first approach is to maintain all users' authentication information in one repository and the second approach is to maintain user authentication information in each domain. Passport and Liberty Alliance are the methods adopting Central Repository (first approach). Under Passport that is developed by Microsoft, users join to Passport server managing by Microsoft and then authenticated by offering his identification. Development goal of Liberty Alliance is to define standard for federated network ID management and for services based on ID. Kerberos, WS-Security, and SAML are the methods attaching authentication information to Web Services messages (second approach). Kerberos is based on ticket. By attaching tickets into SOAP message, a user's authentication information is delivered to the destination domain through intermediaries. WS-Security is defining the mechanism to reflect integrity and reliability to Web Services message. SAML is standard specification for implementing Single Sign-On recommended by OASIS (Organization for Advancement of Structured Information Standards). SAML defines the type of security tokens and provides Web-based Single Sign-On using assertions including tokens.

## 6. Approaches for implementing Single Sign-On

1. The first approach is to maintain a user's authentication list in a central repository.
2. The second approach is to include authentication information for each web service in the initial SOAP message.

### 6.1 The first approach to implementing Single Sign-On

Figure 1 shows the first approach to implement Single Sign-On. In this approach, all users' old IDs are removed and then assigned new IDs from Central Repository. To access Services, a user must use new ID.

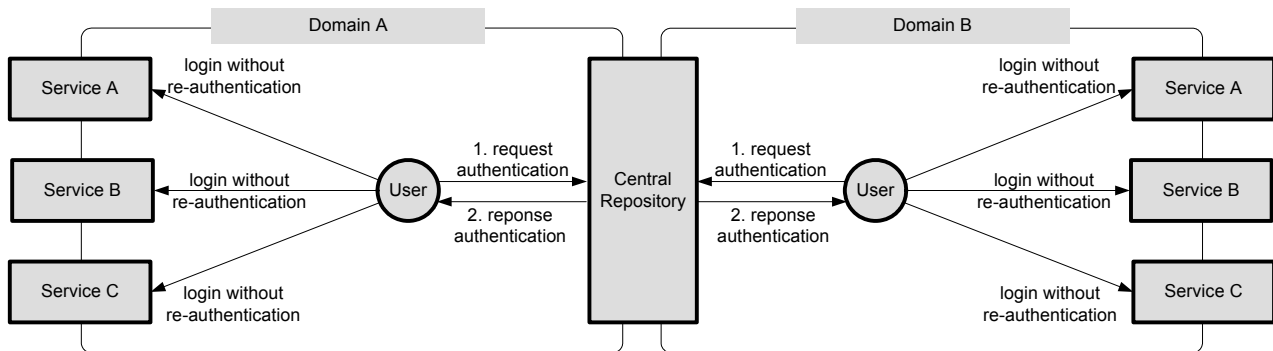


Figure 1 - An approach to implementing Single Sign-On using Central Repository

This approach is proper for single organization. While each organization may lose its control for the administration because all users' information is stored into Central Repository. Also it is difficult to expect the extensibility through this approach. Thus, this approach is not proper to distributed mobile environments such as Web Services, which is set of domain-specific services.

### 6.2 The second approach to implementing Single Sign-On

Figure 2 shows the second approach to implement Single Sign-On. In this approach, all users can use the existing ID to access different Domains without new ID.

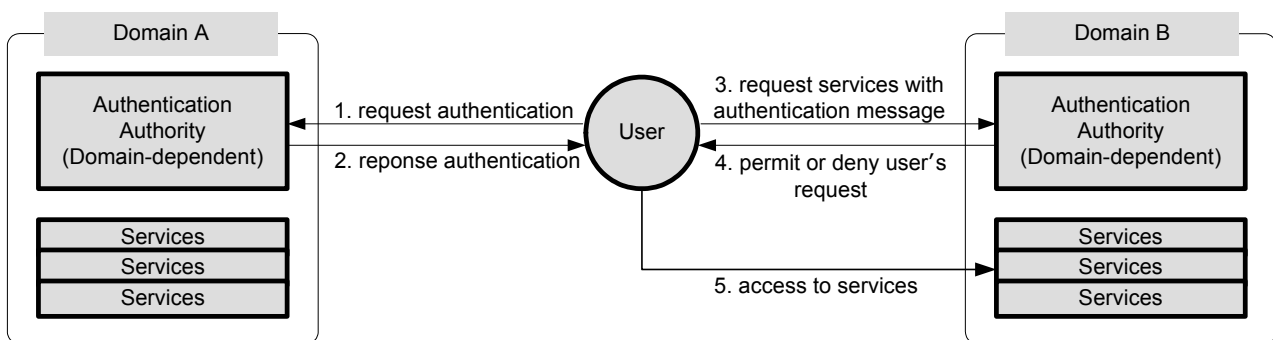


Figure 2 – An approach to implementing Single Sign-On using attached authentication message

This approach is proper for Web Services environment that consists of the domain-dependent distributed services. In such an environment, when a user wants a number of domains, each domain requests authentication to the user. In this case, the user is authenticated by a Domain and then his authentication information is attached to the SOAP message. Under this approach, by transferring the message including authentication information to other domains, the user no longer needs to offer his critical information to each domain.

## 7. Single Sign-On and Access Control Methods in Mobile Web Environment

By comparing two approaches to implement Single Sign-On, we could select an approach suitable for Web Services environments. This section presents Single Sign-On architecture based on the second approach.

To obtain Single Sign-On service, a mobile user must offer his authentication information at least once, and this information is eventually transferred to the Single Sign-On mechanism in a wired service environment. To exchange security information between mobile and wired networks or a domain between other domains, we strongly urge that following considerations should be taken into account:

- The equipment for example a gateway, which operates an appropriate protocol to translate and transfer the user's authentication information among mobile and wired service networks.
- Confidentiality and integrity should be guaranteed during the transfer of the user's authentication information.
- The framework for a Single Sign-On implementation to transfers the user's authentication information is necessary to handle the user authentication mechanism defined in each domain.
- Interoperability should be considered to enables the exchange of domain-specific authentication information to other domains and the reuse of it for user authentication.

According to the considerations, this contribution strongly recommends XML-based Single Sign-On method to extend to wrap around the existing security infrastructures. One of the examples of a widely used framework, which connects mobile and wired service networks, is OMA's WAP (Wireless Application Protocol) gateway, also known as the WAP proxy. The WAP gateway connects the mobile domain and the wired Internet and acts as a protocol gateway to encode and decode content.

This contribution describes an integrated architecture in which a mobile user offers his credential information to the wired service network to obtain user authentication and then access to another domain using this authentication, based on the SAML (Security Assertion Markup Language) that is able to wrap other security infrastructure by defining various acceptable tokens.

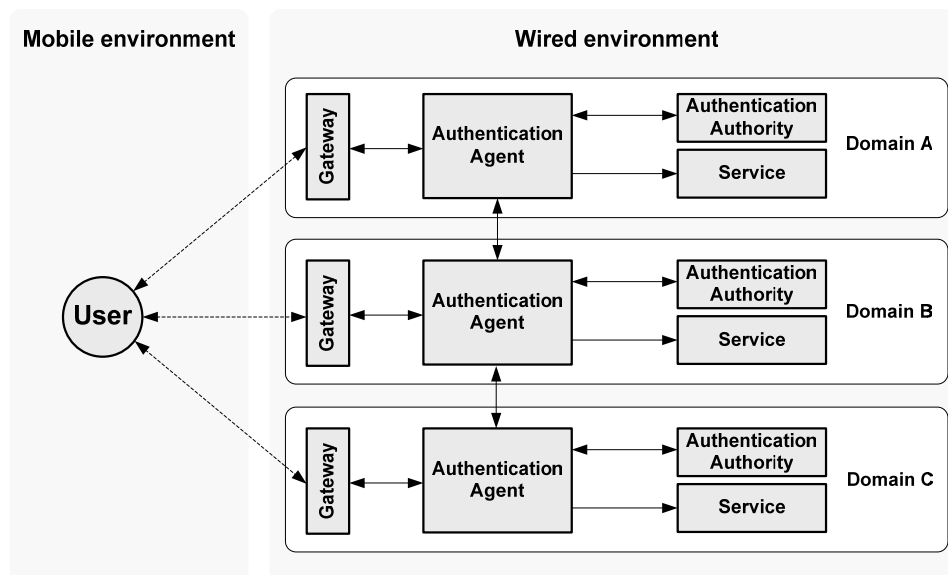


Figure 3 – Architecture for implementing Single Sign-On services

Figure 3 explains the concept of the architecture proposed. A mobile user keys in his or her username and password to the mobile device to access Domain A. This user credential information is transferred to the authentication agent through a gateway that connects the then mobile and wired networks. The authentication agent sends a user authentication request to an authentication authority within Domain A. The authentication authority of Domain A authenticates the user and returns an authentication assertion to the authentication agent of Domain A. The mobile user then has access to a resource in Domain B after successfully getting user authentication from Domain A. Domain B asks for the user's authentication and attribute information to the authentication agent of Domain A; and then the agent sends the authentication and attribute assertion to Domain B. Thus, Domain B performs user authentication by reusing the authentication assertion issued by the authentication authority of Domain A. As the authentication process of Domain B, Domain C reuses the authentication assertion in order to authenticate the user.

The user authentication procedure for this architecture is presented as a sequence diagram in Figure 4. Each box in the diagram denotes an entity involved in the process. Figure 4 explains the messages between entities, applying a user's single sign-on in three domains in which there are mutual trust relationships.

- *Authentication Process #1*, *Authentication Process #2* and *Authentication Process #3* are authenticating a user. When the first authentication of a user is completed successfully, an artifact is generated by the Authentication Agent and assigned to the user.
- In the *Authentication Process #1*, a user must provide ID and password to the system.
- In the *Authentication Process #2*, the user, however, needs not to provide it again because when the user requests authentication to the Authentication Agent of Domain B, the Authentication Agent of Domain A sends the artifact to the Authentication Agent of Domain B and returns it back around from Domain B. The Authentication Agent of Domain A verifies the artifact and sends user authentication information issued by the Authentication Authority within Domain A to the Authentication Agent of Domain B.
- The description about *Authentication Process #* is omitted because *Authentication Process #3* is same with *Authentication Process #2*.

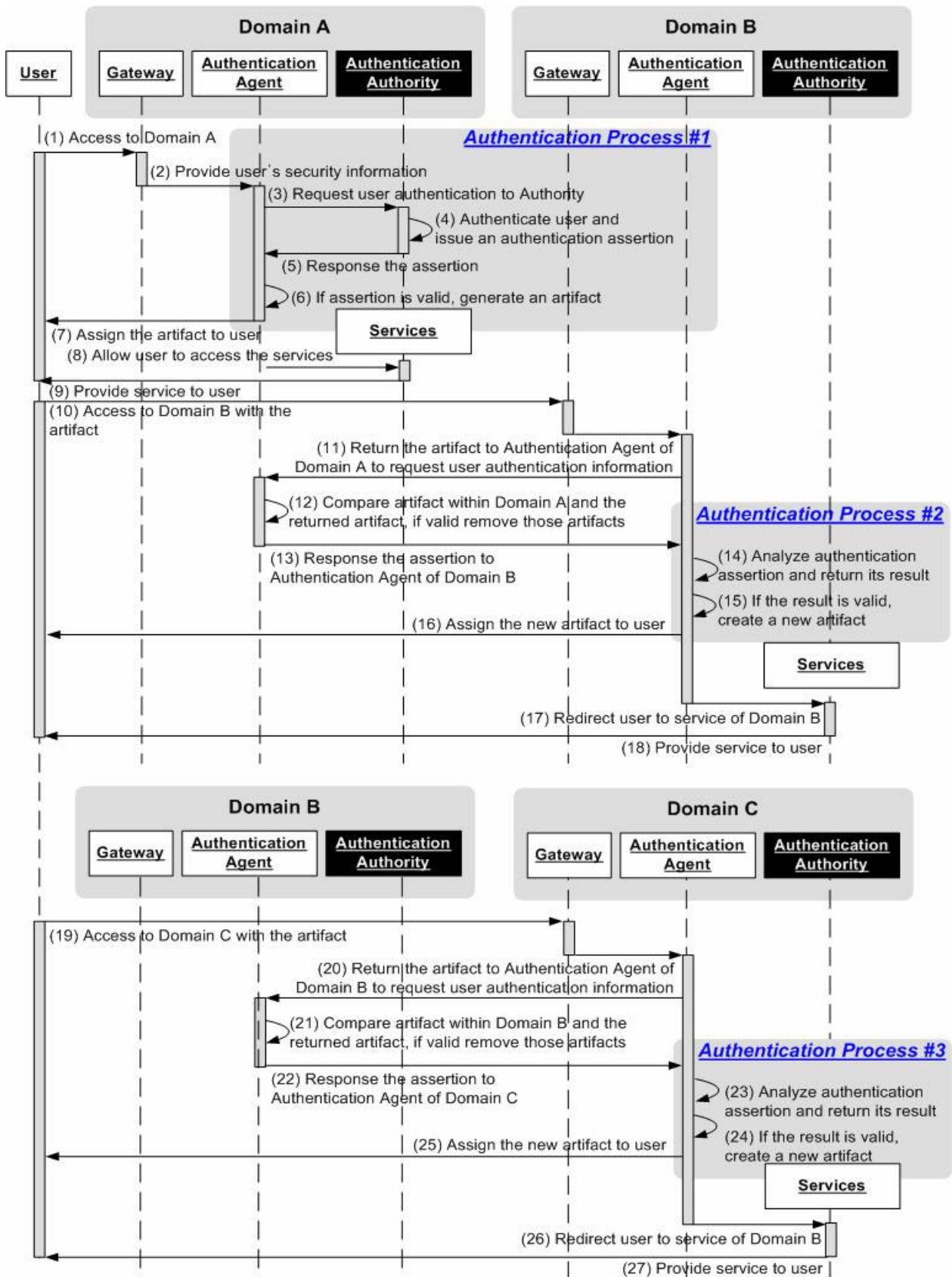


Figure 4 – Sequence diagram of the Architecture we designed



```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  Version="2.0" ID="_00cda300-0d5de-8521-83c5-c2d9f6847b91"
  IssueInstant="2006-03-10T13:33:02Z">
  <saml:Issuer>
    www.etri.re.kr
  </saml:Issuer>
  <saml:Subject>
    <saml:NameID
      Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
        js@etri.re.kr
      </saml:NameID>
    </saml:Subject>
    <saml:Conditions
      NotBefore="2006-03-10T13:33:02Z" NotOnOrAfter="2006-03-10T13:38:02Z" />
    </saml:Conditions>
    <saml:AuthnStatement
      AuthnInstant="2006-03-10T13:33:02Z" SessionIndex="62345344442">
      <saml:AuthnContext>
        <saml:AuthnContextClassRef>
          urn:oasis:names:tc:SAML2.0:ac:classes:Password
        </saml:AuthnContextClassRef>
      </saml:AuthnContext>
    </saml:AuthnStatement>
    <saml:AttributeStatement>
      <saml:Attribute
        Name="jobattribute"
        <saml:AttributeValue>
          <Customer>
            <company>sjcredit</company>
            <email>uuu7@sjcredit.com</email>
          </Customer>
        </saml:AttributeValue>
      </saml:Attribute>
    </saml:AttributeStatement>
  </saml:Assertion>
```

Figure 5 - Assertion with Authentication and Attribute Statement

Figure 5 is an assertion statement issued by the SAML Authority (refers to Step (4) of Figure 4). These messages were verified by a simulation where two domains were constructed with a mutual trust relationship. Figure 5 is showing only core content which is removed the digital signature for entire document or certain element and the XML Encryption of certain element.

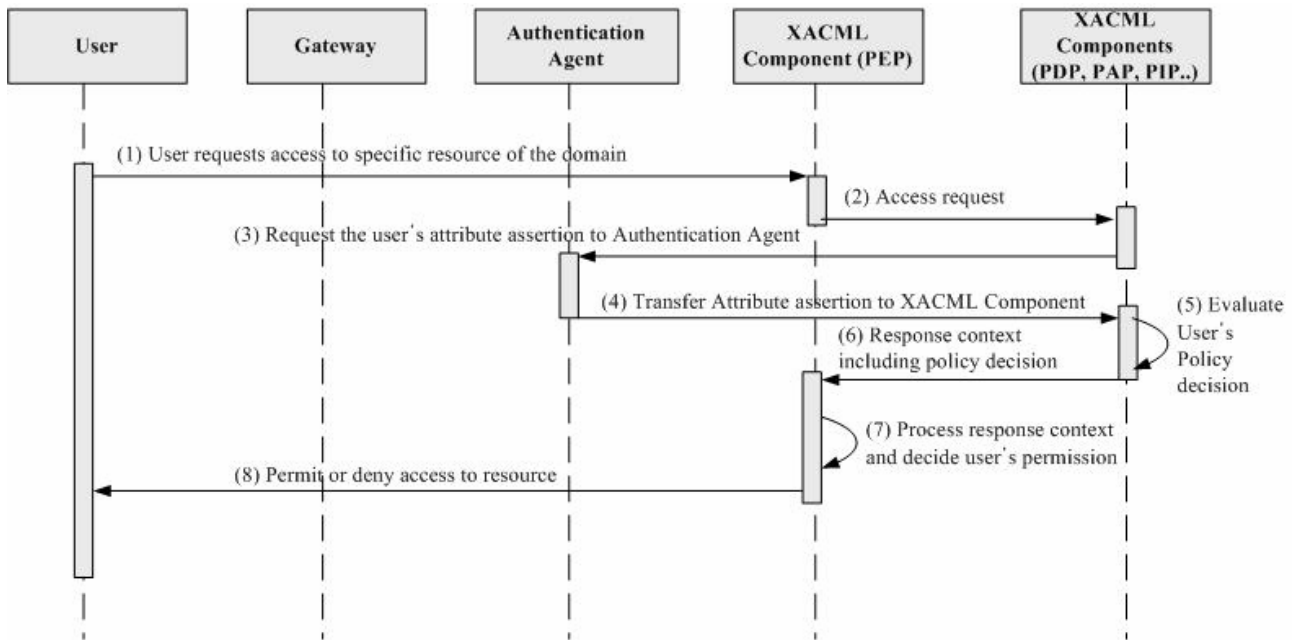


Figure 6 - Sequence Diagram of the Access Control in the Domain

After the user has been directed to the specific service domain (after step 8, 17, 26), access control to the specific resource in the domain may be applied based on XACML. Figure 6 explains the access control mechanism.

### 7.1 Advantages of the proposed methods

The proposed architecture has advantages as follows:

1. The scope of Single Sign-On is extensible because SOAP message acts as a wrapper around the existing security infrastructure.
2. Domain-specific authentication scheme is maintained regardless of other domain-specific authentication schemes because each domain has authentication authority individually.
3. Each domain maintains its control for the administration because various tokens such as X.509 certificate, Kerberos ticket, and hardware token can be attached to SOAP message.
4. In end-to-end security, it is possible to secure critical messages while the message is staying at intermediaries because the message is encrypted.

## 8. Access Control Methods using XACML for UDDI in Web Services

Usually major security concern exists for the user authentication. But critical security problems often occur by an internal user because there is no proper access control policy against an authenticated user. '*Broken Access Control*' is most critical Web application security vulnerabilities. A vulnerable access control mechanism reveals security weaknesses to an attacker who can be understood as an unauthorized internal user, the attacker can perform actions such as read, write, modify, and execute certain resources without any restriction. The attacker can even extort the authority of the whole administration. In Web Services, UDDI is the most important resource to be protected against the attack by unauthorized users. UDDI, which is a core technology of Web Services architecture, is a directory service, which contains information for businesses and services through *businessEntity* and basic data structure of tModel, and offers functionalities of

registration, retrieval and deletion. Service requestor retrieves a service through UDDI and communicates with service provider by generating a client using WSDL (Web Services Description Language). Illegal modification of services within UDDI makes users distrust the services. Therefore, strong access control policy for UDDI should be considered in order to ensure its reliability in Web Services environments.

Figure 7 shows that *XACML Component* decides the access grants for the requests of service provider and service requestor who want to access UDDI. Three major elements of XACML are resource, subject and action. In Figure 1, service provider and service requestor are assigned to subject in XACML, UDDI is assigned to resource and the decision of access grant is assigned to action,

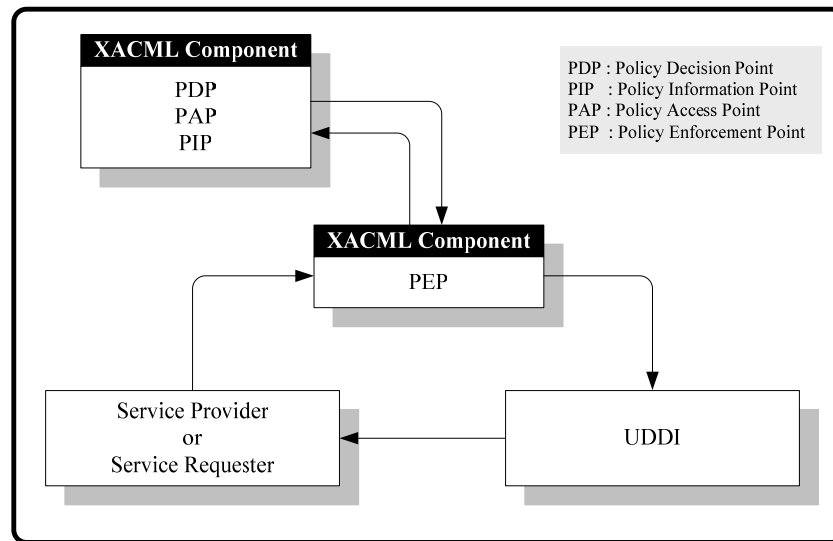


Figure 7 - Architecture for Access Control for UDDI using XACML

Figure 8 shows the procedure of access control for UDDI using XACML. Each step denoted by an arrow and number in the diagram is explained as follows:

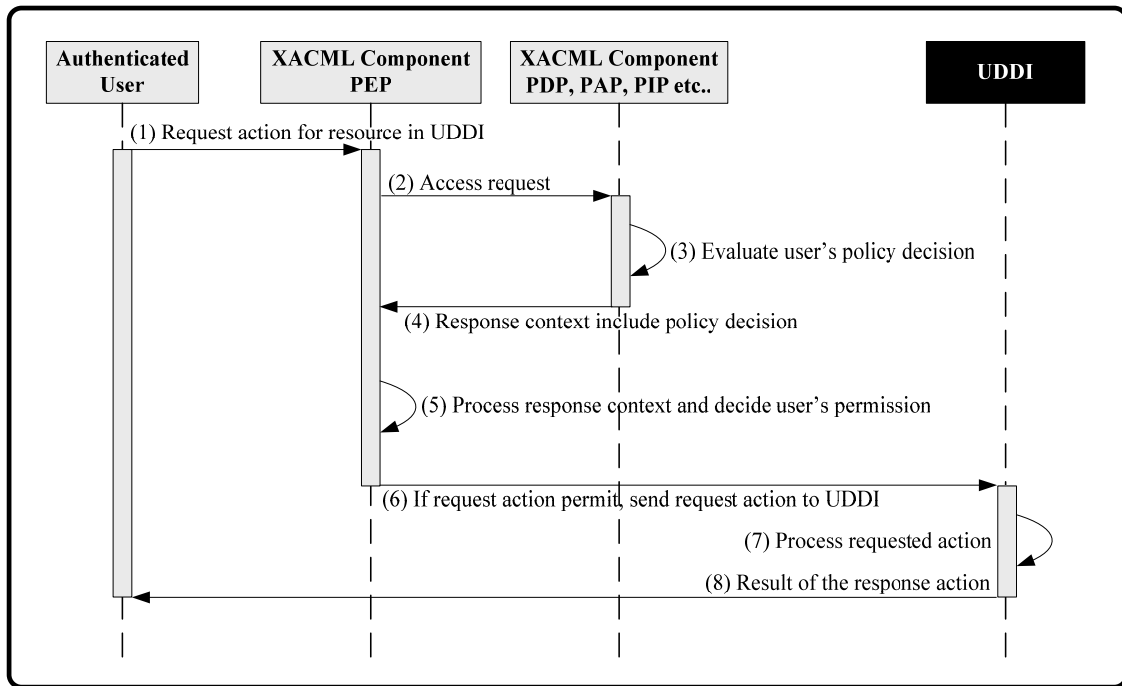


Figure 8 - Sequence Diagram of Access Control for UDDI using XACML

- (1) Authenticated user request action for the resources in UDDI to XACML PEP (Policy Enforcement Point).
- (2) PEP transfer access request of the requested resources to XACML Component.
- (3) XACML Component evaluates the user's Policy.
- (4) Reply context containing determined Policy is transferred to PEP.
- (5) User's access grant is decided after processing the reply context.
- (6) If the requested action is permitted, it is transferred to UDDI.
- (7) UDDI processes the requested action.
- (8) Result of the requested action is sent to the user.

## 9. Attribute Binding for Applying XACML to UDDI

To apply XACML to UDDI, it is necessary to define values of some important attributes, which XACML requires.

### 9.1 Resource Binding

In XACML, *ResourceAttributeDesignator* element identifies *ResourceMatch* or attribute type of resource designated by *Apply* element. To apply XACML to UDDI, following values are defined as shown in Table 4. Here, <attribute> represents the attribute determined by the RegistryObject type.

**Table 4 - Values for ResourceAttributeDesignator**

Resource Attribute	ResourceAttributeDesignator	DataType
Owner	urn:oasis:names:tc:web-service-uddi:3.0:resource:owner	http://www.w3.org/2001/XMLSchema#anyURI
Selector	urn:oasis:names:tc:web-service-uddi:3.0:resource:selector	http://www.w3.org/2001/XMLSchema#string
<attribute>	urn:oasis:names:tc:web-service-uddi:3.0:resource:<attribute>	As defined by attribute definition

### Action Binding

In XACML, *ActionAttributeDesignator* element identifies *ActionMatch* or type of action designated by *Apply* element. In UDDI, Action is UDDI inquiry APIs. To apply XACML to UDDI, following values for *ActionAttributeDesignator* are defined as shown in Table 5.

**Table 5 - Values for ActionAttributeDesignator**

UDDI Action	ActionMatch.ActionAttributeDesignator.AttributeId	Attribute Value
find_business	urn:oasis:names:tc:xacml:1.0:action:action-id	find_business
find_relatedBusinesses	urn:oasis:names:tc:xacml:1.0:action:action-id	find_relatedBusinesses
find_service	urn:oasis:names:tc:xacml:1.0:action:action-id	find_service
find_binding	urn:oasis:names:tc:xacml:1.0:action:action-id	find_binding
find_tModel	urn:oasis:names:tc:xacml:1.0:action:action-id	find_tModel
get_businessDetail	urn:oasis:names:tc:xacml:1.0:action:action-id	get_businessDetail
get_serviceDetail	urn:oasis:names:tc:xacml:1.0:action:action-id	get_serviceDetail
get_bindingDetail	urn:oasis:names:tc:xacml:1.0:action:action-id	get_bindingDetail
get_tModelDetail	urn:oasis:names:tc:xacml:1.0:action:action-id	get_tModelDetail
get_authToken	Urn:oasis:names:tc:xacml:1.0:action:action-id	get_authToken
get_publisherAssertions	Urn:oasis:names:tc:xacml:1.0:action:action-id	get_publisherAssertions

get_assertionStatusReport	Urn:oasis:names:tc:xacml:1.0:action:action-id	get_assertionStatusReport
get_registeredInfo	Urn:oasis:names:tc:xacml:1.0:action:action-id	get_registeredInfo
discard_authToken	Urn:oasis:names:tc:xacml:1.0:action:action-id	discard_authToken
save_business	Urn:oasis:names:tc:xacml:1.0:action:action-id	save_business
save_service	Urn:oasis:names:tc:xacml:1.0:action:action-id	save_service
save_binding	Urn:oasis:names:tc:xacml:1.0:action:action-id	save_binding
save_tModel	Urn:oasis:names:tc:xacml:1.0:action:action-id	save_tModel
delete_business	Urn:oasis:names:tc:xacml:1.0:action:action-id	delete_business
delete_service	Urn:oasis:names:tc:xacml:1.0:action:action-id	delete_service
delete_binding	Urn:oasis:names:tc:xacml:1.0:action:action-id	delete_binding
delete_tModel	Urn:oasis:names:tc:xacml:1.0:action:action-id	delete_tModel
add_publisehrAssertions	Urn:oasis:names:tc:xacml:1.0:action:action-id	add_publisehrAssertions
set_publiserAsseriton	Urn:oasis:names:tc:xacml:1.0:action:action-id	set_publiserAsseriton

## 9.2 Subject Binding

In XACML, *SubjectAttributeDesignator* element identifies *SubjectMatch* or type of action designated by *Apply* element. To apply XACML to UDDI, following values for *SubjectAttributeDesignator* are defined as shown in Table 6.

**Table 6 - Values for SubjectAttributeDesignator**

Subject Attribute	SubjectAttirubuteDesignator	DataType
Id	urn:oasis:names:tc:xacml:1.0:subject:subject-id	http://www.w3.org/2001/XMLSchema#anyURI
Role	urn:oasis:names:tc:web-service-uddi:3.0:subject:role	http://www.w3.org/2001/XMLSchema#string
Group	urn:oasis:names:tc:web-service-uddi:3.0:subject:group	http://www.w3.org/2001/XMLSchema#string

## **10. Rules for Access Control for UDDI using XACML**

To control the access requests of service users and service providers to UDDI of Web Services using XACML, rules must be defined to control access requestors. In addition, various rules must be combined into policy to apply the security of the same target in an XACML document.

The following rules are defined to control the access request of service users and providers to UDDI<sup>1</sup>.

1. Anyone can find business registered in UDDI registry.
2. Service provider identified by authentication system in UDDI registry can do the action of update and deletion for the business and service information that he did registered.
3. Service Broker can perform all kinds of action for the registered business and service information.

Policy applying the above three rules for the security of the same target is represented in XACML document, as shown in Figure 9.

---

<sup>1</sup> The rule for access control can be defined in variety according to variable conditions for resource protection such as characteristics of the target resource for protection and structural environment of the system. The rule defined independently can be combined into a policy to protect specific resources, and also policies can be combined for the same purpose.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:uddi="urn:oasis:names:tc:webse
uddi:3.0" xsi:schemaLocation="C:\uddi\xacml\cs-xacml-schema-policy-01.xsd" policySetId="urn:oasis:names:tc:webservice-uddi:3.0:policy:default-policy"
  PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
- <Policy PolicyId="urn:oasis:names:tc:webservice-uddi:3.0:policy:policyid:permit-all-find_business" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-
combining-algorithm:permit-overrides">
- <Rule RuleId="urn:oasis:names:tc:webservice-uddi:3.0:example:ruleid:1" Effect="Permit">
  <Description>Any Subject can perform read action on any resource.</Description>
  - <Target>
  - <Actions>
  - <Action>
    - <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">find_business</AttributeValue>
      <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="http://www.w3.org/2001/XMLSchema#string" />
    </ActionMatch>
  </Action>
</Actions>
</Target>
</Rule>
</Policy>
- <Policy PolicyId="urn:oasis:names:tc:webservice-uddi:3.0:policy:policyid:permit-ServiceProvdierr" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:permit-overrides">
- <Rule RuleId="urn:oasis:names:tc:webservice-uddi:3.0:example:ruleid:2" Effect="Permit">
  <Description>A Subject with role of service provider can perform save_business and delete_business on resources owned by them.</Description>
  - <Target>
  - <Subjects>
    - <Subject>
      - <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Service Provider</AttributeValue>
      </SubjectMatch>
    </Subject>
  </Subjects>
  - <Resources>
    - <Resource>
      - <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">business_entity</AttributeValue>
        <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:webservice-uddi:3.0:resource"
          DataType="http://www.w3.org/2001/XMLSchema#string" />
      </ResourceMatch>
    </Resource>
  </Resources>
  - <Actions>
  - <Action>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">save_business</AttributeValue>
    <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string" />
  </ActionMatch>
  </Action>
  - <Action>
    - <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">delete_business</AttributeValue>
      <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="http://www.w3.org/2001/XMLSchema#string" />
    </ActionMatch>
  </Action>
</Actions>
</Target>
- <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
  - <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
    <AttributeSelector RequestContextPath="//uddi:find_relatedBusinesses/uddi:businessKey/text()"
      DataType="http://www.w3.org/2001/XMLSchema#string" />
  </Apply>
</Condition>
</Rule>
</Policy>
- <Policy PolicyId="urn:oasis:names:tc:webservice-uddi:3.0:policy:policyid:permit-UDDIAdministrator-any-action"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
- <Rule RuleId="urn:oasis:names:tc:webservice-uddi:3.0:example:ruleid:2" Effect="Permit">
  <Description>A Subject with role of RegistryAdministrator can perform any action on any resource.</Description>
  - <Target>
  - <Subjects>
    - <Subject>
      - <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">UDDIAdministrator</AttributeValue>
      </SubjectMatch>
    </Subject>
  </Subjects>
  - <Resources>
    - <Resource>
      - <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">business_entity</AttributeValue>
        <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:webservice-uddi:3.0:resource"
          DataType="http://www.w3.org/2001/XMLSchema#string" />
      </ResourceMatch>
    </Resource>
  </Resources>
</Target>
</Rule>
</Policy>
</PolicySet>
```

Figure 9 – XACML document for the Access Control in UDDI