Implementation Experiences On IHE XUA and BPPC[1]

December 5, 2006

Tuncay Namlı and Asuman Dogac
Software Research and Development Center
Middle East Technical University
Ankara, Turkey

The most up-to-date version of this document is available from
http://www.srdc.metu.edu.tr/publications

[1] This work is supported in part by the European Commission, eHealth Unit (http://ec.europa.eu/information_society/activities/health/index_en.htm) through the 027074 Saphire Project (http://www.srdc.metu.edu.tr/webpage/projects/saphire/) and by the Scientific and Technical Research Council of Turkey (TUBITAK) through the Project No. EEEAG 105E133.

List of Figures

List of Acronyms

BPPC - IHE Basic Patient Privacy Consent Profile
ECP - Enhanced Client/Proxy
EHR - Electronic Healthcare Record
IHE - Integrated Healthcare Enterprise
IDP - Identity Provider
PAP - Policy Administration Point
PDP - Policy Decision Point
PEP - Policy Enforcement Point
PIP - Policy Information Point
POP - Proof-of-Possession Token
PWP - IHE Personal White Pages Profile
RBAC - Role Base Access Control
SAML - OASIS Security Assertion Markup Language Specification
SP - Service Provider
SSO - Single Sign-On
STS - Security Token Service
TLS - Transport Layer Security
WSS - Web Service Security Specification
XACML - OASIS Extensible Access Control Markup Language Specification
XDS - IHE Cross Enterprise Document Sharing Profile
XUA - IHE Cross Enterprise User Authentication Profile

1 Overview

We have implemented SAML Enhanced Client and Proxy (ECP) profile [SAMLProfiles] for Cross Enterprise Document Sharing (XDS) Retrieve transaction. During the implementation we have extensively analyzed SAML Web Single Sign-On [SAMLProfiles] and ECP profiles. In this document we share our implementation experiences on these profiles and give some recommendations for the Cross Enterprise User Authentication (XUA) profile [XUA2006]. Furthermore, we have analyzed some WS standardization efforts for which the XUA profile has referenced. We provide a summary of these WS efforts and some recommendations.

Cross Enterprise User Authentication (XUA) profile aims to provide a way for user authentication across enterprises. In fact, the objective of the profile is to transfer user identity information in cross-enterprise transactions rather than specifying how to authenticate or authorize the users for the requested services. The profile mentions 'auditing' and 'access decisions' as two main processes that will use the user identity information.

The XUA profile specifies OASIS Security Assertion Markup Language (SAML) v2 standards as the backbone of the system. In the profile, SAML assertions are referenced as security tokens carrying user identity and authentication information across IHE actors. Currently, SAML Web

SSO and ECP profiles are referenced to specify the process flow and transactions. On the other hand, IHE considers changing the transactions to the web services which is also mentioned in the last version of the profile. Therefore, the newer versions of XUA profile most probably will refer WS standardization efforts for the identity management transactions in which SAML assertions are used as security tokens.

2 Executive Summary

A summary of issues mentioned in this document is as follows:

- In the Identity Federation architectures (both in the Web Service efforts and SAML specifications), the trusted intermediaries are the basic actors to federate the identity and trust among the service providers and the service clients. Therefore, the XUA profile should specify a model describing the relationships between service providers, service clients and the trusted intermediaries (Identity Providers in SAML specifications and Security Token Service in WS-Trust model) in an affinity domain. Furthermore, it should specify how this model can be extended for the federation of affinity domains.

- In the ECP Profile, the Identity Provider (IDP) should be able to identify the subject (principal) in order to give an assertion about the subject. Therefore, either there should be prior established security context between the Identity Provider (IDP) and the Enhanced Client (ECP) or the context should be established with initiating a fresh authentication. However, the ECP Profile does not specify any method or communication flow for this purpose. In our implementation we use cookies. In this respect, in order to provide interoperability, XUA profile should specify standard methodologies.

- In the XUA profile, the content of the SAML assertions are not specified exactly. In the ECP profile, SAML assertions can carry authentication and attribute statements. However, the requested attributes can not be specified in SAML AuthnRequest message which is sent by the Service Provider. The ECP Profile can be extended in this respect. Furthermore, the SAML attribute query mechanisms do not handle more complex attribute queries (e.g. asking a functional role of a professional for a patient).

- Some of the procedures, elements and attributes are left optional in SAML Profiles. By using the SAML Metadata specification, actors using the SAML framework can define their choice for the optional things. In addition, these actors can define their requirements (e.g. required attributes for authorization) and preferences regarding the SAML profiles. The SAML Metadata specification can be recommended for the actors implementing the XUA profile to define the related metadata needed in SAML framework.

- After getting the SAML Assertion, the Service Provider can use the authentication statement in the assertion to establish a security context with the subject. If such a context is not established, same process should be repeated for each request for the service (e.g. a user may perform several XDS queries). SAML specifications do not specify a mechanism for this purpose. On the other hand, WS-SecureConversation can be used for web service transactions to establish security context in the Service Provider side.

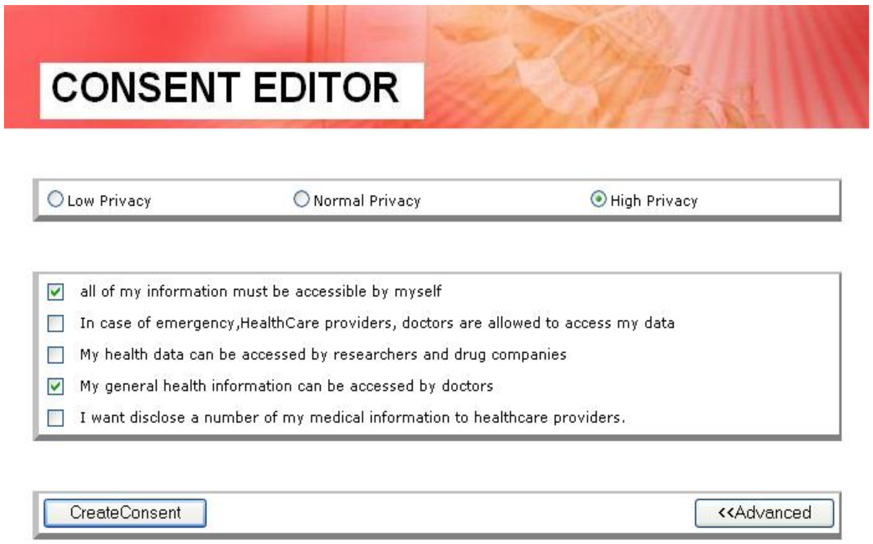
- In the XUA profile while profiling the cross-enterprise user authentication, the WS-Trust specification will be the main element for web service transactions. The WS-Trust specification defines the basic building blocks (like SAML Core specification does) to construct a trust model. However, it needs further profiles (like SAML Profiles) which define the usage of these building blocks to handle specific use-cases (e.g. single sign on).

- BPPC does not restrict the content of the Privacy Consent Policies. Therefore, the implementations of the access control mechanisms will be manual and specific to the Privacy Consents defined in the Affinity Domain. On the other hand, the implementation of the mechanisms will be very easy if IHE selects a machine processable access policy standard for the Privacy Consent Policies. The XACML standard seems to be very suitable for this purpose. It can provide all functionalities for the access control systems mentioned in the BPPC profile. In addition, it also supports more complex functionalities for future refinements and the profiles.

3 The Implementation Scenario

Our implementation is realized through a scenario which is based on an XDS Affinity Domain where the “Electronic Health Record (EHR)” of a patient is shared between two healthcare institutes. The implementation concentrates on the “XDS Retrieve Transaction” and provides the “Cross User Authentication” and “Patient Consent” based authorization services on this transaction. For “Cross User Authentication”, SAML ECP profile is implemented. The authorization service is implemented based on Role Base Access Control (RBAC) model using the OASIS Extensible Access Control Markup Language (XACML) standard [XACML2.0].

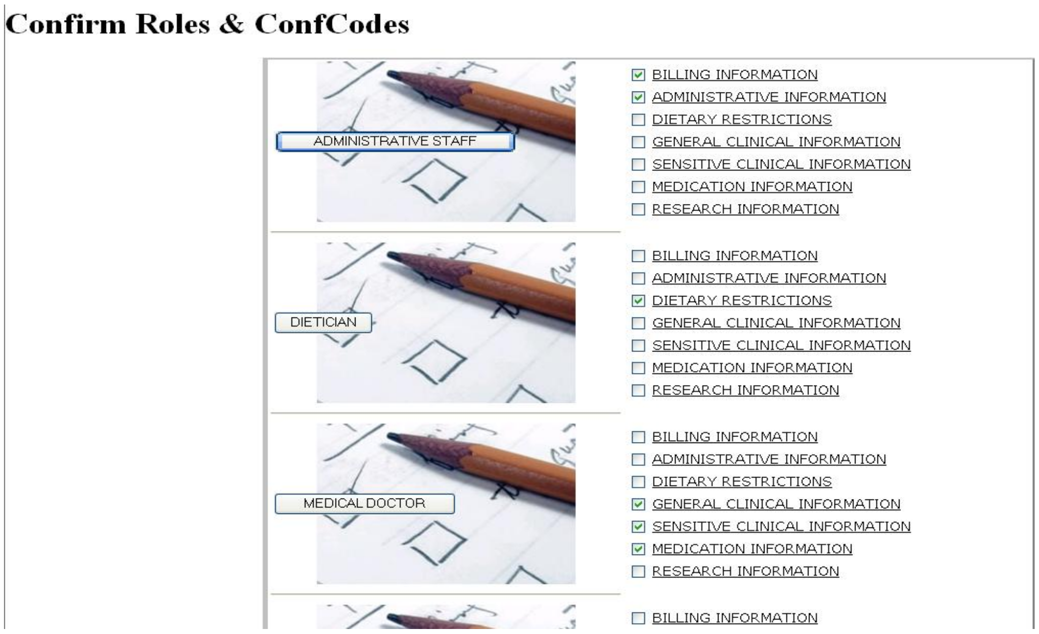
The scenario starts within a healthcare Institute A where a patient is treated for his medical problems. When the patient is discharged from the institute, several medical documents were already produced including the discharge summary of the patient. The healthcare institute obtains the patient consent. The patient consent sets the rules for sharing and the use of these documents. We have provided a web base consent editor tool as shown in the Figure 1. The patient constructs his consent by using this tool. The tool produces an XACML policy which corresponds to consent rules. The policy is based on RBAC model where the healthcare professional roles and classification of documents in terms of sensitivity have been already specified in the XDS affinity domain. The Consent Editor is designed in three modes. The Figure 1 illustrates the basic mode of the Consent Editor which provides some basic choices for the patient. The choices are defined by simple sentences. The editor binds these sentences to the appropriate rules which are defined as configurations of the editor. The advance mode shown in the Figure 2 provides an interface to match the user roles and classification of documents.


CONSENT EDITOR

Low Privacy   Normal Privacy   [✓ High Privacy]

- [✓] all of my information must be accessible by myself
- [ ] In case of emergency,HealthCare providers, doctors are allowed to access my data
- [ ] My health data can be accessed by researchers and drug companies
- [✓] My general health information can be accessed by doctors
- [ ] I want disclose a number of my medical information to healthcare providers.

Figure 1 Consent Editor – Basic Mode



Confirm Roles & ConfCodes

ADMINISTRATIVE STAFF
- ☑ BILLING INFORMATION
- ☑ ADMINISTRATIVE INFORMATION
- ☐ DIETARY RESTRICTIONS
- ☐ GENERAL CLINICAL INFORMATION
- ☐ SENSITIVE CLINICAL INFORMATION
- ☐ MEDICATION INFORMATION
- ☐ RESEARCH INFORMATION

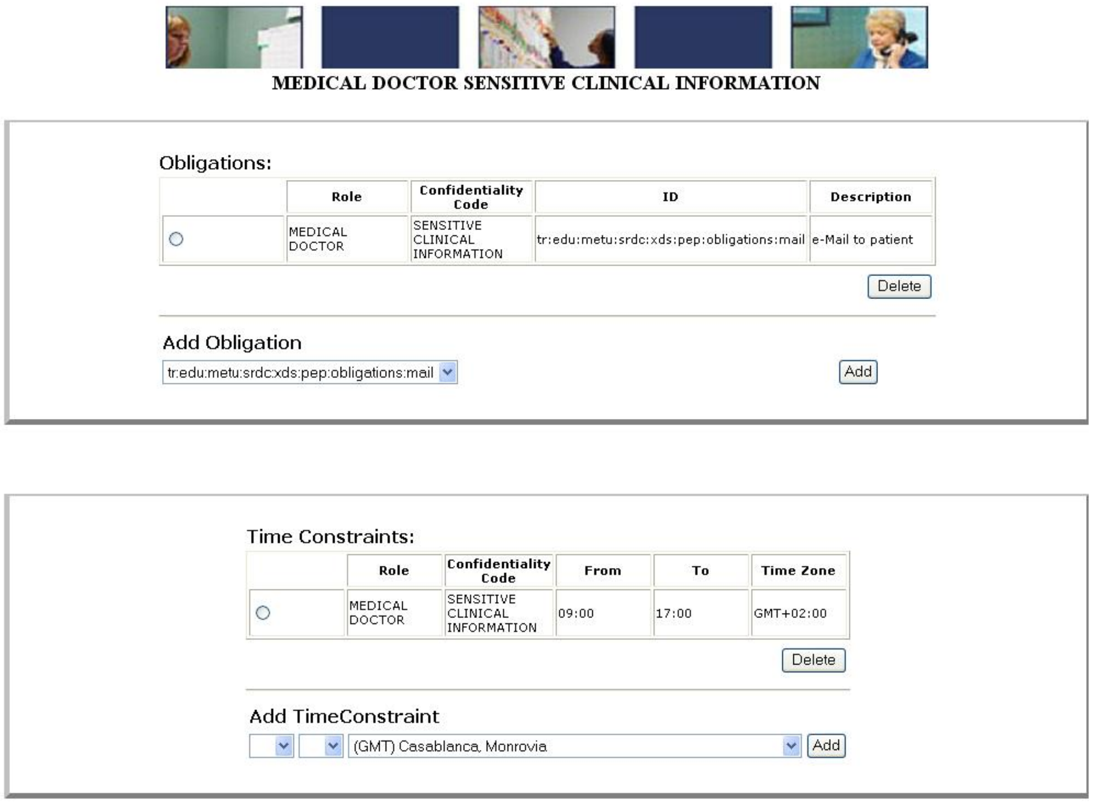
DIETICIAN
- ☐ BILLING INFORMATION
- ☐ ADMINISTRATIVE INFORMATION
- ☑ DIETARY RESTRICTIONS
- ☐ GENERAL CLINICAL INFORMATION
- ☐ SENSITIVE CLINICAL INFORMATION
- ☐ MEDICATION INFORMATION
- ☐ RESEARCH INFORMATION

MEDICAL DOCTOR
- ☐ BILLING INFORMATION
- ☐ ADMINISTRATIVE INFORMATION
- ☐ DIETARY RESTRICTIONS
- ☑ GENERAL CLINICAL INFORMATION
- ☑ SENSITIVE CLINICAL INFORMATION
- ☑ MEDICATION INFORMATION
- ☐ RESEARCH INFORMATION

- ☐ BILLING INFORMATION

Figure 2 Consent Editor - Advance Mode

In addition, the patients can set some restrictions, for example time ranges for access and mail obligation (e.g. when the document is accessed a mail must be sent to the patient). The Figure 3 shows the last mode which provides an interface to add constraints and obligations for the role-sensitivity matching.



MEDICAL DOCTOR SENSITIVE CLINICAL INFORMATION

Obligations:

	Role	Confidentiality Code	ID	Description
○	MEDICAL DOCTOR	SENSITIVE CLINICAL INFORMATION	tr:edu:metu:srdc:xds:pep:obligations:mail	e-Mail to patient

Add Obligation

tr:edu:metu:srdc:xds:pep:obligations:mail

Time Constraints:

	Role	Confidentiality Code	From	To	Time Zone
○	MEDICAL DOCTOR	SENSITIVE CLINICAL INFORMATION	09:00	17:00	GMT+02:00

Add TimeConstraint

(GMT) Casablanca, Monrovia

Figure 3 Obligations & Constraints

After the policy is obtained, it is sent to XDS Repository by the institute. While sending the consent through the “XDS Provide and Register Document” transaction, some of the attributes in the metadata should be set. For example, class code is set as ‘Consent’ which marks the document as consent. After sending the consent, the medical documents are also sent. In their metadata, ‘confidentialityCode’ entry specifies their sensitivity (e.g. General Clinical Information).

Now assume that, a healthcare professional Mr. X from Institute B needs to access these documents. First, Mr. X authenticates himself to the institute B’s information system. When the security context is established for Mr. X, the information about this context is sent to the Identity Provider which the Institute B is registered in the affinity domain. The Identity Provider also provides some attribute values (as an Attribute Authority Service) like the user role in the institute. Then, Mr. X queries the XDS Registry, selects the Discharge Summary Document and requests it from the XDS Repository. Before the actual transaction takes place, the SAML ECP profile is executed. The Identity Provider provides the required assertions. Then required attributes are retrieved from the Identity Provider by using the SAML Assertion Query/Request [SAMLProfiles] and SAML Attribute Profiles [SAMLProfiles]. After all these transactions, XDS Repository finds the consent of the patient related with the requested document. The authorization service uses this consent which is in XACML format and the attributes obtained from the Identity Provider and decide on the access right for Mr. X. Then according to the authorization decision the document is sent to Mr. X.

4 Trust Model

4.1 Trust Model in an Affinity Domain

Both in SAML and WS-Trust [WS-Trust] specifications, the trust model depends on the delegation of trust to the intermediary trust brokers. These intermediary actors are called X-Identity Provider in IHE XUA profile, Identity Provider (IDP) in SAML and Security Token Service (STS) in WS-Trust model. In this way, the providers of services that need user authentication (user’s identity) only need to trust the claims of these trusted entities.

Several identity management models can be constructed by using these intermediaries. However, considering the IHE affinity domain concept, the model would be as given in Figure 4.

In this model, any service provider should have trust relationships with all X-Identity Provider actors. This trust should be in two ways; that is, any service provider should trust the claims of the X-Identity Provider that is located in the affinity domain and any X-Identity Provider should ensure that the entity which the claim will be sent to is one of the authorized service providers in the affinity domain. On the other hand, one X-Identity provider can serve claims of several institutes. In this model, using a single X-Identity Provider for the whole affinity domain can also be considered.

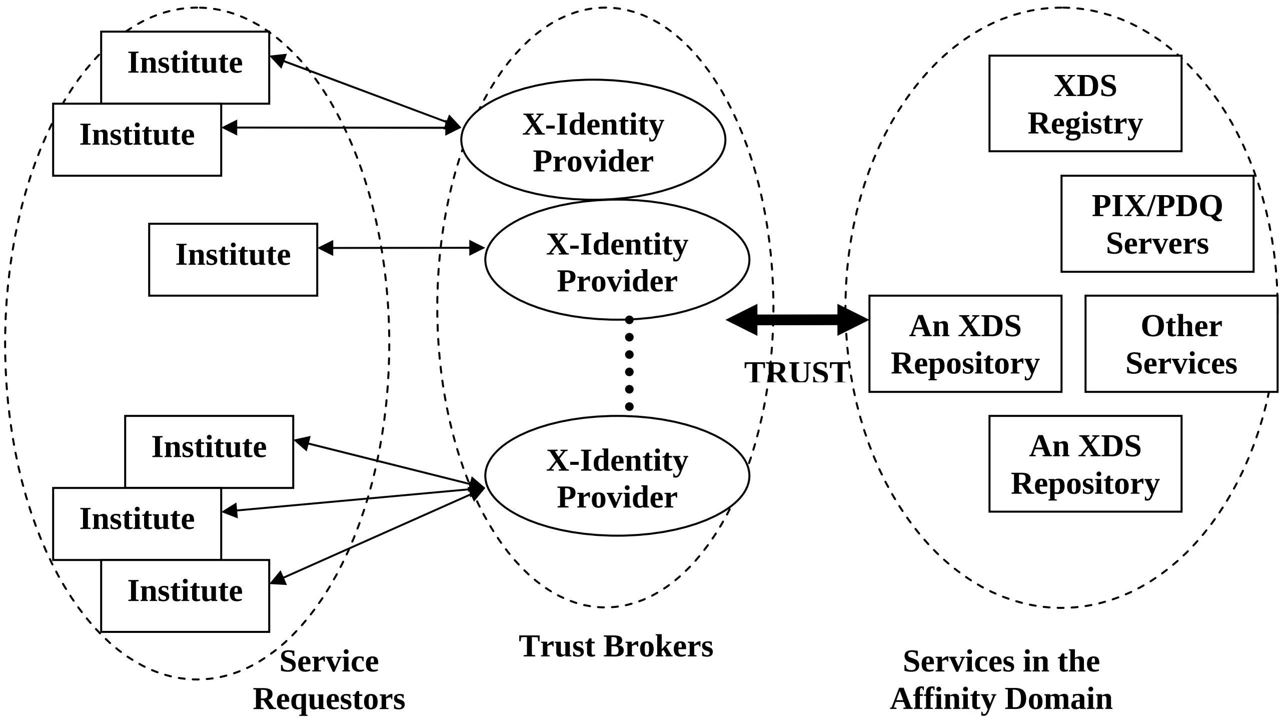


Figure 4 A Trust Model for IHE Affinity Domain

In this abstract model, any service (XUA enabled) request within the affinity domain will be accompanied by some XUA transactions including the three XUA actors; namely X-Service Provider, X-Identity Provider and X-Service User. All these three actors communicate with each other to share the user identity. The arrows in Figure 5 represent the communication paths between these actors. The order of communication paths is important and should be restricted for interoperability. In fact, one of the important functionality of SAML Profiles is restricting the message flow to certain patterns. WS-Trust specification also mentions such future profiles to give restrictions over communication flows.

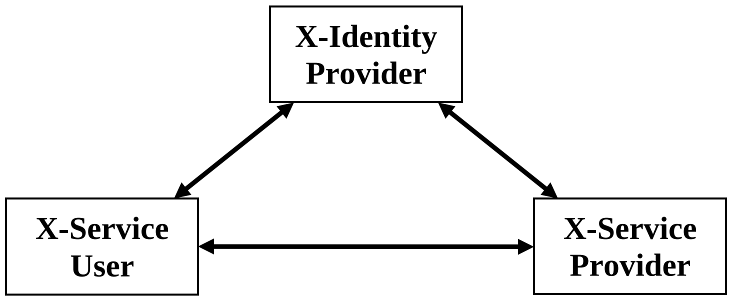


Figure 5 XUA Actors

XUA profile provides two process flows which are “Pre-Generated Assertions” and “Post-Generated Assertions”. In order to use “Pre-Generated Assertions”, clients should be aware of the service provider preferences and restrictions. The X-Service User gets the assertion according to the known X-Service Provider preferences from the X-Identity Provider and sends this assertion together with service request to X-Service Provider. In the Post-Generated Assertions case, after the request of the X-Service User, the X-Service Provider requests an assertion by defining its preferences as a response.

To finalize the IHE XUA profile, one flow for Pre-Generated Assertions and one flow for Post-Generated Assertions should be specified or bound to some standard profiles. These profiles may be specific to healthcare and defined by IHE in cooperation with standard bodies or selected from the existing SAML profiles and WS standardization efforts.

4.2 Trust Model for Federated Affinity Domains

When we also consider the federation of affinity domains, the model given in the Figure 4 can be extended as shown in Figure 6. In this case, X-Service Provider in the Affinity Domain B does not have a direct trust relationship with the X-Identity Provider in the Affinity Domain A. Therefore, claims of this identity provider for X-Service User are not acceptable by X-Service provider. We propose to extend the trust chain to include the Identity Providers in both of the affinity domains. In this way it becomes possible to manage the identity of the user across affinity domains. SAML and WS-Trust specifications both mention and supplement such scenarios. In fact, the trust between the X-Identity Providers does not need to be a direct trust.

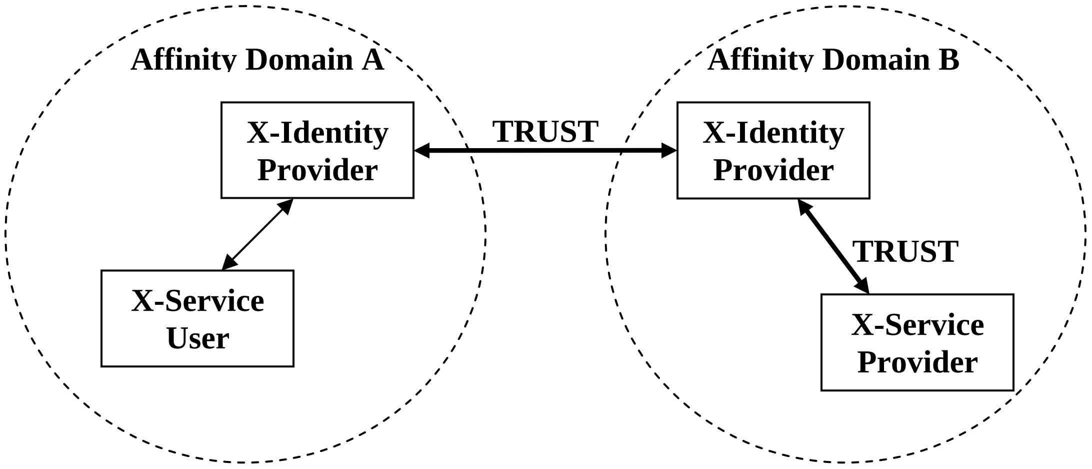


Figure 6 Trust in Federation of Affinity Domains

5 SAML and XUA

In this section we give an example describing the whole cycle of ECP profile and discuss its capabilities and limitations from IHE viewpoint. Web SSO profile is also included in these discussions.

5.1 SAML ECP Profile

"An enhanced client or proxy (ECP) is a system entity that knows how to contact an appropriate identity provider, possibly in a context-dependent fashion, and also supports the Reverse SOAP (PAOS) binding" [SAMLBindings]. From the definition it is clear that ECP communicates directly with its Identity Provider (IDP). The whole message flow is shown in the Figure 7 [SAMLTechnical Overview].

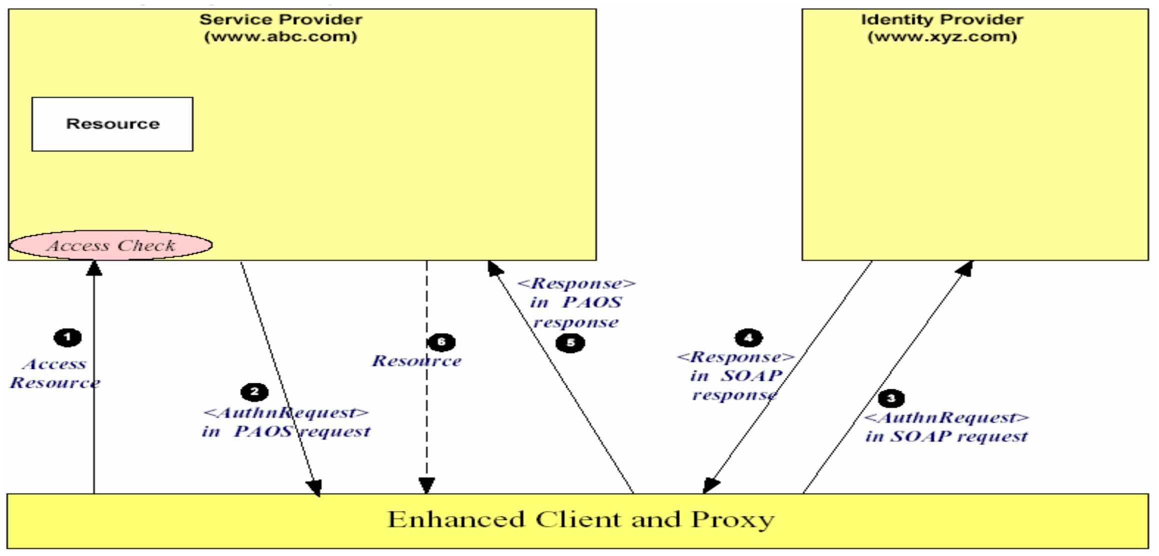

Service Provider
(www.abc.com)
Resource
Access Check
Identity Provider
(www.xyz.com)
1 Access Resource
2 <AuthnRequest> in PAOS request
3 <AuthnRequest> in SOAP request
4 <Response> in SOAP response
5 <Response> in PAOS response
6 Resource
Enhanced Client and Proxy

Figure 7 ECP Message Flow

We give an example message flow from our implementation to demonstrate some of the details of the ECP Profile.

5.1.1 SAML AuthnRequest

The Figure 8 shows the HTTP GET request for the XDS Retrieve transaction from an ECP actor. The only changes in this message are the PAOS header which shows that the ECP actor has implemented the ECP Profile and use the PAOS binding.

```
GET /xdsServices/xdsRep/f5aabb42-bd47-419f-b4f9-f48b18574000.xml HTTP/1.1
Accept: text/html; application/vnd.paos+xml
PAOS: ver='urn:liberty:paos:2003-08';'urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp'
Cache-Control: no-cache
Pragma: no-cache
User-Agent: Java/1.5.0_05
Host: 144.122.230.23:7070
Connection: keep-alive
Content-type: application/x-www-form-urlencoded
```

Figure 8 XDS Retrieve HTTP Request

When the X-Service Provider (SP) receives this request, it checks the PAOS header to ensure that the requester supports SAML ECP Profile. After verification, SP constructs a SAML *AuthnRequest* message from its configuration and preferences. We define the SP's preferences through some configuration files. SAML provides SAML Metadata specification [SAMLMetadata] in this respect. However the API, openSAML 2.0 [openSAML], that we are using has not implemented Metadata section yet so we use our simple configuration xml files. A brief description of SAML Metadata specification and some examples are given in the next sections.

The objective of SAML *AuthnRequest* element is to request an authentication statement from a trusted IDP about a user. In this respect, it is suitable for the XUA profile. In this request, SP can declare its preferences and restrictions over the authentication and the response that it will receive. The Figure 9 shows an example SAML *AuthnRequest* element.

```
<samlp:AuthnRequest
        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        AssertionConsumerServiceURL="https://144.122.230.23:8443/xdsServices/xdsRep/AssertionConsumer"
        ForceAuthn="false"
        ID="_2zC8NK7kjZNvqaT"
        IsPassive="true"
        IssueInstant="2006-10-03T06:29:03.450Z"
        ProviderName="SRDC-XDS" Version="2.0">
        <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" SPProviderID="sp"/>
        <saml:Conditions xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
        NotOnOrAfter="2006-10-03T06:32:03.450Z"/>
        <samlp:RequestedAuthnContext Comparison="exact">
                <saml:AuthnContextClassRef
               xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
                        urn:oasis:names:tc:SAML:2.0:ac:classes:Password
                </saml:AuthnContextClassRef>
        </samlp:RequestedAuthnContext>
        <samlp:Scoping>
                <samlp:IDPList>
                        <samlp:IDPEntryLoc="http://144.122.230.23:9091/identityprovider/IdentityProvider"
Name="SRDC-Care2x" ProviderID="idp"/>
                        <samlp:GetComplete>/TrustedIDPList.xml</samlp:GetComplete>
                </samlp:IDPList>
        </samlp:Scoping>
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
              ………
       </ds:Signature>
</samlp:AuthnRequest>
```

Figure 9 SAML AuthnRequest

The most important element in this *AuthnRequest* is the *RequestedAuthnContext* element. It defines the SP's restrictions for the user authentication in the Service User side. In our example, we state that authentication method must be the method defined with the unique URI *urn:oasis:names:tc:SAML:2.0:ac:classes:Password*. SAML has defined such methods in the SAML Authentication Context specification [SAMLAuthContext]. It also allows declaring such contexts by giving reference to xml schemas.

SAML *Scoping* element gives the information about IDPs which the SP trusts. In our example scenario, this is all the Identity providers in the same affinity domain of SP.

In Figure 9, the signature of SP is shown which the IDP will use to authenticate and verify the integrity of the message. ECP profile does not mandate the signature for *AuthnRequest* element but it is strongly recommended. However, it states that IDP must verify any *AssertionConsumerServiceURL* which must be the real endpoint of the SP whose identifier is given in the *Issuer* element. In our configuration, IDP has a list of trusted SPs with their IDs, assertion consumer URLs and certificates. This information is used to authenticate SP, check the *AssertionConsumerServiceURL* attribute and verify the integrity of the message.

AuthRequest element is defined in the SAML Core specification [SAML 2.0]. ECP Profile sets the restrictions about the message binding and set some more restrictions over the XML structure as in the other SAML profiles. The sample message with PAOS binding is shown in Figure 10. The message has two required headers; *paos:Request* and *ecp:Request*. These headers are for the use of ECP and removed by ECP before forwarding the message to IDP. The *messageID* attribute of *paos:Request* header is used to correlate the message with the SOAP response. Nevertheless, since the *AuthnRequest* element has also a unique id which is used for the same purposes, this attribute in the *paos:Request* header is optional. In our implementation this attribute is set by the SP and used as session ID when waiting for the authentication response. By using this attribute, we provide the message correlation in SOAP header level.

SP sends the message including the *AuthnRequest* to the ECP. ECP process the headers, remove them and forwards the message to its IDP with SAML SOAP binding. As described in the Trust Model section, we assume that each ECP has one IDP which serves assertion about the users in the system that ECP is working. ECP profile does not specify anything about how to choose the IDP. Only the selected IDP should be trusted by the SP. As mentioned before, the list of trusted IDPs are given in the *ecp:Request* header. In addition to these, the value of *responseConsumerURL* attribute of *paos:Request* header should be stored. This value gives SP's endpoint URL of the services which processes the response from the ECP.

The ECP profile recommends the use of SSL 3.0 [SSL3.0] or TLS 1.0 [TLS1.0] in order to maintain confidentiality and integrity of the whole message. In fact, the integrity of the SAML elements inside the message like *AuthnRequest* element is protected by signatures however the SOAP headers also should be protected. This is already mentioned in the current version of XUA profile [XUA2006] by recommending the use of IHE ATNA profile [IHE-ATNA] together with XUA. IHE ATNA uses the TLS 1.0 to provide node authentication and provide confidentiality and integrity of the whole communication line between the nodes. In this way, the SOAP headers are also protected as requested in the ECP profile.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Header>
                <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
                        SOAP-ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
                        SOAP-ENV:mustUnderstand="1" messageID="7J6iylkrkM6KB”
responseConsumerURL="https://144.122.230.23:8443/xdsServices/xdsRep/AssertionConsumer"
                        service="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"/>
                <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
                IsPassive="1" ProviderName="SRDC-XDS"
                        SOAP-ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
                        SOAP-ENV:mustUnderstand="1">
                        <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
SPProviderID="sp"/>
                        <samlp:IDPList xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
                                <samlp:IDPEntry Loc="http://144.122.230.23:9091/identity-
provider/IdentityProvider" Name="SRDC-Care2x" ProviderID="idp"/>
                                <samlp:GetComplete>/TrustedIDPList.xml</samlp:GetComplete>
                        </samlp:IDPList>
                </ecp:Request>
        </SOAP-ENV:Header>
        <SOAP-ENV:Body>
                <samlp:AuthnRequest>
                        ....
                </samlp:AuthnRequest>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 10 SAML AuthnRequest Message PAOS Binding

5.1.2 Identification of Principal by IDP

The IDP should identify the user (subject) for whom the SP wants authentication statement. The ECP Profile optionally allows the use of SAML *Subject* element inside the *AuthnRequest* element. By using *Subject* element, the SP can state the principal for whom it requests authentication statement (assertions). However, in order to do this it needs to identify the principal from the service request. SAML does not specify anything for this; neither IHE has such specifications for its services. As seen from the Figure 9, we do not include such a *Subject* element. In this case SAML Core [SAML2.0] specification states that presenter of the message is assumed to be the subject. Nevertheless, in both cases, IDP should identify the principal. This identification is not only discovering an id or a name of the subject but establishing a security context with the principal. This step is mentioned but not included in the ECP profile scope. It only states that IDP must establish the identity of principal by any means; either it may start a new act of authentication or may reuse existing authentication session.

This issue is very critical in an IHE affinity domain. If X-Identity Providers are planned to be individual external entities as in our model given in Figure 4 (because self assertion scenarios can not represent real life as discussed in open issues [XUA2006]), the methodology and the interface between IDP and ECP while establishing the security context should be clearly specified. XUA profile [XUA2006] has also mentioned this relationship between the corresponding actors *User Authentication Provider* (which is assumed to be located in ECP)*, X-Assertion Provider* (XUA terminology for Identity Provider). Nevertheless, the profile also states that this relationship is out of scope that is how X-Assertion Provider gets the authentication information to create an assertion is left to the implementations.

In the light of these discussions, we describe how our implementation handles this step. In our implementation framework, when a user is authenticated to the web interface (simple user-password authentication) for the *Document Consumer* actor which is actually our ECP, the authentication information is sent to the IDP. This transformation is simple HTTP Post and the content of the transaction is not bound to any standard. When IDP receives the authentication information it generates a session for the user (stores authentication info) and sends a cookie to establish the security context. The cookie is used by the ECP for the transactions between the IDP so that the IDP can identify the principal and generates the assertion from the authentication information in the session. The SAML has provided an Implementation Guideline [SAMLImplGuideline] which discusses how cookies can be used in session state maintenance and security context establishment.

In summary, standard mechanisms need to be specified for the relationship between IDP and ECP to exchange authentication information of user. SAML Assertion Query Request Protocols can be easily used for this purpose with some context management mechanisms like cookies. However, some one way protocol may be needed in which authentication statement is directly sent without a query or a request. Furthermore, WS-SecureConversation specification [WS-SecureConversation] has the main objective of establishing security context and may be used if the web service transactions are used. More details are provided on this issue in the following sections.

5.1.3 IDP Response

When IDP identifies the principal, we can assume that any information that is needed for the response is ready. The next step for IDP is to check if the authentication preferences of SP given in *RequestedAuthnContext* element are satisfied by the ECP when authenticating the user to the ECP's system. From this authentication information, IDP generates an *AuthenticationStatement*. The Figure 11 shows the SAML *Response* element including the *AuthenticationStatement* which gives the authentication instant and method.

SAML *Response* element has an *InResponseTo* attribute which is actually the ID of AuthnRequest sent by SP. This attribute provides message correlation. The *Status* element shows the status of the response; *urn:oasis:names:tc:SAML:2.0:status:Success* means IDP has the authentication information of the user which is suitable for the SP's preferences. This attribute can take other values to describe the problems that can occur while checking user authentication,

authentication of SP, signature verifications, processing of the message. Some of these values we used in our implementation are as follows;

- urn:oasis:names:tc:SAML:2.0:status:VersionMismatch
- urn:oasis:names:tc:SAML:2.0:status:RequestDenied
- urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext
- urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal

```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" ID="_4o9XZa7wzfwsjKu"
InResponseTo="_2zC8NK7kjZNvqaT" IssueInstant="2006-10-03T06:30:05.778Z" Version="2.0">
        <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" SPProviderID="idp"/>
        <samlp:Status>
                <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
        </samlp:Status>
        <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" ID="_lNt7I5qek4IWxRc"
IssueInstant="2006-10-03T06:30:05.778Z" Version="2.0">
                <saml:Issuer SPProviderID="idp"/>
                <saml:Subject>
                        <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified">doe</saml:NameID>
                        <saml:SubjectConfirmation
Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                                <saml:SubjectConfirmationData
InResponseTo="_2zC8NK7kjZNvqaT" NotOnOrAfter="2006-10-03T06:33:05.778Z"
Recipient="https://144.122.230.23:8443/xdsServices/xdsRep/AssertionConsumer"/>
                        </saml:SubjectConfirmation>
                </saml:Subject>
                <saml:Conditions>
                        <saml:AudienceRestriction>
                                <saml:Audience>sp</saml:Audience>
                        </saml:AudienceRestriction>
                </saml:Conditions>
                <saml:AuthnStatement AuthnInstant="2006-10-03T06:15:05.778Z"
SessionNotOnOrAfter="2006-10-03T07:15:05.778Z">
                        <saml:SubjectLocality Address="127.0.0.1"/>
                        <saml:AuthnContext>
                                <saml:AuthnContextClassRef>
                                        urn:oasis:names:tc:SAML:2.0:ac:classes:Password
                                </saml:AuthnContextClassRef>
                        </saml:AuthnContext>
                </saml:AuthnStatement>
                <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                        ...
                </ds:Signature>
        </saml:Assertion>
</samlp:Response>
```

Figure 11 SAML Response

ECP profile allows more than one SAML *Assertion* element inside the *Response*. In our scenario, we use only one assertion which is used to give the authentication statement. The *Assertion* must include a *Subject* which gives a name identifier for the subject of the assertion. In our implementation, we use user ID of the user in the ECP's system. In real life, if we consider the health domain, this should be a health professional identifier which is unique for the affinity domain. SAML provides *Name Identifier Mapping Profile* which proposes a solution when two parties (SP and IDP) do not use the same identifiers for the user. WS-Federation has also defined some use cases for this purpose.

The *Subject* element must include a *SubjectConfirmation* element. The element is used by the relying party (SP in our case) to confirm that the request or message came from a system entity (ECP in our case) that is associated with the subject of the assertion [SAMLProfiles].

The *Method* attribute of *SubjectConfirmation* element gives an identifier of the method which SP must use to confirm that the subject of the assertion is actual subject of the request. SAML has defined three such methods which are used in the SAML profiles.

ECP Profile mandates the use of *Bearer* method as it is used in the example shown in Figure 11. The *SubjectConfirmationData* element states that the bearer (carrier) of this assertion can be confirmed to be the real subject only if the assertion is delivered in a message sent to *https://144.122.230.23:8443/xdsServices/xdsRep/AssertionConsumer* before *2006-10-03T06:33:05.778Z*.

The other methods can optionally be used in ECP profile. *Holder-of-Key* method is more secure. In this method the IDP put some information about the key of the subject into the *SubjectConfirmationData* element. This information can be the name of the key or the whole key data (e.g X509 Public certificate). The requesting party (ECP in our case) signs the assertion with the private key of the user before sending it to the relying party (SP in our case). SP use the information inside the *SubjectConfirmationData* (it may be the whole certificate and can be directly use as a public key for signature verification) to find the key and verify the signature of the subject by using this key. If verification is successful, subject is assumed to be confirmed.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Header>
                <ecp:Response
                        xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
                        AssertionConsumerServiceURL="https://144.122.230.23:8443/xdsServices/xds
Rep/AssertionConsumer"
                        SOAP-ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
                        SOAP-ENV:mustUnderstand="1"/>
        </SOAP-ENV:Header>
        <SOAP-ENV:Body>
                <samlp:Response>
                        ...
                </samlp:Response
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 12 IDP Response with SOAP Binding

Finally IDP must sign all assertions inside the *Response*. It may also sign the *Response* element. These signatures will be verified by SP in order to be sure about the integrity of the Response. IDP sends the *Response* to the ECP with SOAP binding. As seen from Figure 12, *ecp:Response* header must be used in the SOAP message. The *AssertionConsumerServiceURL* attribute is set by IDP by using the value of *AssertionConsumerServiceURL* attribute in *AuthnRequest* sent by SP. ECP check this value with the *ResponseConsumerURL* in the *paos:Request* header in the authentication request message sent by SP. If the values are not the same, there is a possibility of man-in-the-middle attack that is some unauthorized external entity behaves like SP to obtain the assertions about the subject.

5.1.4 Processing Response

ECP forwards the SAML *Response* element to SP with PAOS binding. The Figure 13 shows the SOAP message that ECP sends to SP. *paos:Response* header is required according to the ECP Profile. If *messageID* attribute is used in *paos:Request* header in authentication request message (as we use in our example), *refToMessageID* attribute is also required in *paos:Response* header for message correlation.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Header>
                <paos:Response
                        xmlns:paos="urn:liberty:paos:2003-08"
                        SOAP-ENV:actor="http://schemas.xmlsoap.org/soap/actor/next"
                        SOAP-ENV:mustUnderstand="1"
                        refToMessageID="7J6iylkrkM6KB "/>
        </SOAP-ENV:Header>
        <SOAP-ENV:Body>
                <samlp:Response>
                        ...
                </samlp:Response>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 13 ECP forwards the Response – PAOS binding

The first thing that SP must do is verifying the signatures present on the SAML *Assertions* and *Response* elements. SP should also authenticate the IDP that is it should check if IDP is in the trust list of SP. However this is not mentioned explicitly in the ECP profile.

In our implementation, we identify the IDP from the *Issuer* element in the *Assertions* (which is a required element according to ECP Profile). As a requirement of the ECP Profile, there should be a trust relationship between the IDP and SP. Therefore as discussed in the SAML AuthnRequest section, like IDP, SP has also a list of trusted IDPs (IDPs in the affinity domain as in our model) with their IDs (Provider ID used in the SAML messages), endpoint addresses (IDP service endpoint) and certificates. To authenticate IDP, we get the identified IDP's certificate and check if it is equal to the certificate in the signature.

SP should check the *SubjectConfirmation* element to confirm that the assertion is related to the given subject. However, regardless of the subject confirmation method, SP must check some attributes: the *Recipient* attribute should be equal to the SP's own assertion consumer URL and *InResponseTo* attribute should be equal to ID of the *AuthnRequest*. SP then should check the optional elements inside the *Conditions* element.

After all of these checks, the *AuthnStatement* element can be used to establish a security context with the user (given in the *Subject*). However, the *SessionOnOrNotAfter* attribute must be considered for the life of this established security context.

5.1.5 Discussions on SAML SSO Profiles

SAML ECP Profile is more suitable for IHE XUA rather than SAML Web SSO Profile since the latter is totally browser based. On the other hand, ECP Profile does not mention pre-generated assertions which are one of the given types of the assertions in IHE XUA profile. If an X-Service User wants to use pre-generated assertions (which is called unsolicited responses in SAML profiles), the requirements of the SP about the assertion must be known. One way to achieve this is using the SAML Metadata specification for the SPs and IDPs. SAML Metadata is briefly described in the following sections. Another way is putting strong restrictions over the assertion content. In any case, there is a need for a profile (not browser based, can be a modification of ECP) for pre-generated assertions.

Another issue is the communication between the IDP and ECP as discussed in the above sections. If IDPs are considered as separate entities in the affinity domain, the interface (how to communicate the authentication information, how to initiate fresh authentications) should be defined by profiles.

IHE services may also need authorization mechanisms on the requested resources or services. In order to conclude such access control decisions, the authorization mechanisms need some attributes about the user (user role, email, etc). SAML provides Attribute Profiles for this purpose. In our scenario, we obtain such attributes by using the SAML Attribute profiles after obtaining the authentication information by using the ECP. Nonetheless, IHE may need a combined profile in which attribute values can be gathered from IDPs during the authentication.

ECP Profile facilitates this by using SAML Metadata. In the authentication request message, SP gives an identifier with *AttributeConsumingServiceIndex* attribute. This value is used by IDP to access the SAML Metadata document of SP (Publishing and resolution of Metadata documents are described in SAML Metadata [SAMLMetadata] specification). In this Metadata document, SP defines the required attributes for its authorization service. IDP finds the values of these attributes and puts them in the SAML *Response* element as *AttributeStatements*. In this architecture, using SAML Metadata will become a must for SPs. Therefore, some other way may be provided which can be the extension of *AuthnRequest* element in the way that it can include SAML *Attribute* elements to query for attribute values (that is SP can ask for attribute values).

Reporting the failures during the execution of the profile is also very important. SAML provides this information within the SAML *Status* element in its transactions. The *StatusCode* element gives the identifier for the status. These identifiers are defined in SAML Core specification and define some basic possible statuses and failures in SAML transactions. However, the objective of the *Status* element is to report the statuses or failures to the requestor of the assertion (SP in ECP), not reporting them to users. Therefore failure alternatives in the selected profiles, the mapping of the SAML status code values to these failures and the way to report them to users should be identified.

5.2 SAML Attribute Profiles

SAML Attribute Profiles give the specifications about how to name attributes and how to compare them. We use SAML XACML Attribute Profile since we use the attribute values for access control decision in our scenario. LDAP Attribute Profile, UUID Attribute Profile and DCE/PAC Attribute Profile are other important attribute profiles in SAML. SAML Assertion Query/Request Profile gives the specification of requesting attributes from an Attribute Authority (AA).

The Figure 14 shows the SAML *AttributeQuery* element which SP sends to IDP in the body of a simple SOAP message. The *Subject* element gives the identifier of the subject that the attributes are requested for. Then each *Attribute* element states the name and name format of the requested attributes. As in the ECP profile, the communicating parties (SP as attribute requester and IDP as attribute authority in our case) must authenticate each other. Message correlation is handled by the *ID* and *InResponseTo* attributes as seen from the Figure 14 and the Figure 15.

```
<samlp:AttributeQuery xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        ID="nARKciu2Hxlng53"
        IssueInstant="2006-10-03T06:31:06.778Z" Version="2.0">
        <saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" SPProviderID="sp"/>
        <saml:Subject xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
                <saml:NameID>doe</saml:NameID>
```

```
		</saml:Subject>
		<saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" FriendlyName="User
Role" Name="urn:oasis:names:tc:XACML:2.0:example:attribute:role"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
		<saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" FriendlyName="User
Email" Name="urn:oasis:names:tc:XACML:2.0:example:attribute:email"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
		<saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" FriendlyName="User
Name" Name="urn:oasis:names:tc:XACML:2.0:example:attribute:subject-name"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"/>
		<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
			...
		</ds:Signature>
</samlp:AttributeQuery>
```

Figure 14 SAML Attribute Query

The SAML *Response* for the attribute query is shown in the Figure 15. The *AttributeStatement* element inside the *Assertion* provides the attribute values for the requested attributes. In order to respond such attribute queries, the Attribute Authority (attribute authority is IDP in our scenario) must have the ability of resolving the attribute from the attribute name and providing the value for the resolved attribute. The Attribute Authority can use the SAML Metadata to provide the list of attributes it can handle. The attribute names must be common and unique for both requester and the Attribute Authority sides for attribute resolution.

IHE IT Infrastructure Planning Roadmap 2004-2009 Beyond [IHERoadmap] has mentioned future profile candidates for enterprise and cross enterprise RBAC systems. In addition, Basic Patient Privacy Consent (BPPC) in PCC framework profile defines a way of using more than one privacy policy in an affinity domain. SAML Attribute profiles may play a major role in these profiles. They can either be used independently or with combination of XUA as discussed in the section 5.1.5. However, there are some open issues which should be decided by these profiles while using the SAML Attribute Federation mechanism.

```
<samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol" ID="_b2tgTl8l7s71AO7"
InResponseTo="nARKciu2Hxlng53"
IssueInstant="2006-10-03T06:31:07.247Z" Version="2.0">
		<saml:Issuer xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" SPProviderID="idp"/>
		<samlp:Status>
			<samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
		</samlp:Status>
		<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" ID="_87y4RjTPbwpFr0e"
IssueInstant="2006-10-03T06:31:07.247Z" Version="2.0">
			<saml:Issuer SPProviderID="idp"/>
			<saml:Subject>
				<saml:NameID>doe</saml:NameID>
			</saml:Subject>
			<saml:AttributeStatement>
				<saml:Attribute FriendlyName="User Role"
					Name="urn:oasis:names:tc:XACML:2.0:example:attribute:role"
					NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
					<saml:AttributeValue
xmlns:xs="http://www.w3.org/2001/XMLSchema"	xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="xs:String">DIETICIAN</saml:AttributeValue>
				</saml:Attribute>
				<!-- Other Attributes-->
			</saml:AttributeStatement>
```

```
                    <ds:Signature>
                              ...
                    </ds:Signature>
          </saml:Assertion>
</samlp:Response>
```

Figure 15 SAML Attribute Statement

First of all, these profiles should determine;

- Required and optional attributes for the profile,
- From which entity (Attribute Authorities) these attributes can be gathered; trusted entities like IDP, self assertions, other services like Personal White Pages (PWP) directory,
- How these attributes can be registered to the attribute authorities,
- How the attributes can be named (selecting appropriate SAML Attribute Profile).

Some attributes need to be updated very often rather than other static attributes like demographic information for a healthcare professional. Functional role of a healthcare professional on a patient is such an attribute which can not be simply stored forever. The information about these attributes is located in the information system of the healthcare enterprise and this information should be opened to outside by some services. In such situations using self assertions seem suitable. Another problem is that the SAML AttributeQuery can not handle querying such dynamic attributes. While requesting the attributes, only name of the attribute is stated in the SAML Attribute element. However, the attribute can depend on a three sided relationship which is subject-attribute-object relationship rather that subject-attribute relationship. For the above case, we should somehow state the patient identifier (like we give the subject identifier) in AttributeQuery to request the functional role of the professional on the patient.

5.3 SAML Metadata

SAML Metadata [SAMLMetadata] specification defines a way for SAML entities to agree and share system identifiers, endpoints, supported profiles, certificates and keys. It also defines a way to publish and find these metadata definitions. In this section we present the metadata definitions for the IDP and the SP that can be used in our ECP implementation scenario.

The Figure 16 shows the metadata of the IDP. The root element is the *EntityDescriptor* with the *entityID* attribute stating the unique identifier of the entity in the domain in which all these SAML issues are performed. The ds:*Signature* element is just to protect the integrity of the metadata definition. The last element, *Organization*, presents the basic information about the entity. The other elements under the root element, the *IDPSSODescriptor* and the *AttributeAuthorityDescriptor*, present the roles that the entity can play in the SAML profiles and protocols.

The *IDPSSODescriptor* element states the details of the role of IDP in the SSO profile (ECP profile in our case). For example, the *WantAuthenticationRequestsSigned* attribute states that SP must sign the *AuthnRequest* elements which is left optional in the ECP profile. The IDP's certificate which will be used to sign the Assertions in the IDP's response message is given with the *KeyDescriptor* element. The SP may use the information in this element to retrieve the IDP's certificate for signature verifications and IDP authentication. In the service elements, the *SingleSignOnService* in this example shows the details of the services provided by a specified role. The endpoint and binding of the service is given by the *Location* and *Binding* attributes of the element. As mentioned in the section 5.1.5, if it is desired to transfer attribute values during the ECP Profile, IDP may state the supported attributes within this element as it is presented in the *AttributeAuthorityDescriptor*.

```
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   entityID="idp">
   <ds:Signature>...</ds:Signature>
   <IDPSSODescriptor WantAuthnRequestsSigned="true"
            protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
            <KeyDescriptor use="signing">
               <ds:KeyInfo><ds:X509Data><ds:X509Certificate> ...
</ds:X509Certificate></ds:X509Data></ds:KeyInfo>
            </KeyDescriptor>
            <SingleSignOnService
               Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
               Location="http://144.122.230.23:9091/identity-provider/IdentityProvider"/>
              <!--Attributes supported  in the ECP Response -->
              <!--Same as the Attributes in AttributeAuthorityDescriptor-->
   </IDPSSODescriptor>
   <AttributeAuthorityDescriptor
            protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
            <KeyDescriptor use="signing">
               <ds:KeyInfo><ds:X509Data><ds:X509Certificate> ...
</ds:X509Certificate></ds:X509Data></ds:KeyInfo>
            </KeyDescriptor>
            <AttributeService
               Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
               Location="http://144.122.230.23:9091/identity-provider/AttributeAuthority"/>
            <saml:Attribute
               NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
               Name="urn:oasis:names:tc:XACML:2.0:example:attribute:role"
               FriendlyName="UserRole">
               <saml:AttributeValue>ADMINISTRATIVESTAFF</saml:AttributeValue>
               <saml:AttributeValue>DIETICIAN</saml:AttributeValue>
               <saml:AttributeValue>MEDICAL DOCTOR</saml:AttributeValue>
               <saml:AttributeValue>NURSING STAFF</saml:AttributeValue>
               <saml:AttributeValue>PHARMACIST</saml:AttributeValue>
               <saml:AttributeValue>RESEARCHER</saml:AttributeValue>
            </saml:Attribute>
            <saml:Attribute
               NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
               Name="urn:oasis:names:tc:XACML:2.0:example:attribute:email"
               FriendlyName="UserEmail"/>
            <saml:Attribute
               NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
               Name="urn:oasis:names:tc:XACML:2.0:example:attribute:subject-name"
               FriendlyName="UserName"/>
   </AttributeAuthorityDescriptor>
   <Organization>
            <OrganizationName xml:lang="en">METU Identity Provider </OrganizationName>
            <OrganizationDisplayName xml:lang="en">
               Ankara - IHE Affinity Domain - METU Identity Provider
            </OrganizationDisplayName>
            <OrganizationURL xml:lang="en">http://www.srdc.metu.edu.tr</OrganizationURL>
   </Organization>
</EntityDescriptor>
```

Figure 16 Metadata of Identity Provider

The *AttributeAuthorityDescriptor* element describes the Attribute Authority service given as an example in the section 5.2. The *Attribute* elements inside the *AttributeService* define the supported attributes by IDP with their names and possible values.

```
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   entityID="https://ServiceProvider.com/SAML">
   <ds:Signature>...</ds:Signature>
   <SPSSODescriptor AuthnRequestsSigned="true"
            protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
            <KeyDescriptor use="signing">
               <ds:KeyInfo>
                        <ds:X509Data><ds:X509Certificate> ... </ds:X509Certificate></ds:X509Data>
               </ds:KeyInfo>
            </KeyDescriptor>
            <AssertionConsumerService index="0"
               Binding="urn:oasis:names:tc:SAML:2.0:bindings:PAOS"
               Location="https://144.122.230.23:8443/xdsServices/xdsRep/AssertionConsumer"/>
            <AttributeConsumingService index="0">
               <ServiceName xml:lang="en">IHE XDS Retrieve Document</ServiceName>
               <RequestedAttribute
                        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                        Name="urn:oasis:names:tc:XACML:2.0:example:attribute:role"
                        FriendlyName="UserRole">
                        <saml:AttributeValue>ADMINISTRATIVESTAFF</saml:AttributeValue>
                        <saml:AttributeValue>DIETICIAN</saml:AttributeValue>
                        <saml:AttributeValue>MEDICAL DOCTOR</saml:AttributeValue>
                        <saml:AttributeValue>NURSING STAFF</saml:AttributeValue>
                  <saml:AttributeValue>PHARMACIST</saml:AttributeValue>
                  <saml:AttributeValue>RESEARCHER</saml:AttributeValue>
               </RequestedAttribute>
               <RequestedAttribute
                        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                        Name="urn:oasis:names:tc:XACML:2.0:example:attribute:email"
                        FriendlyName="UserEmail"/>
               <RequestedAttribute
                        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
                        Name="urn:oasis:names:tc:XACML:2.0:example:attribute:subject-name"
                        FriendlyName="UserName"/>
            </AttributeConsumingService>
   </SPSSODescriptor>
   <Organization>
            <OrganizationName xml:lang="en">METU XDS Repository</OrganizationName>
            <OrganizationDisplayName xml:lang="en">
               Ankara IHE Affinity Domain - METU XDS Repository
            </OrganizationDisplayName>
            <OrganizationURL xml:lang="en">http://www.srdc.metu.edu.tr</OrganizationURL>
   </Organization>
</EntityDescriptor>
```

Figure 17 Metadata of Service Provider

The metadata of the SP is illustrated in the Figure 17. The *AssertionConsumerService* gives the endpoint and binding of the assertion consumer at the SP side. Furthermore, the *AttributeConsumingService* element defines the required attribute values for the XDS Retrieve transaction. These attributes are used for auditing and authorization services in our implementation. As discussed before, these attributes can be either included in the IDP's response in ECP Profile or requested with the Assertion Query/Request Profile (as it is done in our implementation). The *AttributeValue* elements within the UserRole attribute states the possible values for the user role attribute.

6 WS Standards and XUA

In this section we briefly describe security related WS standards already mentioned in the IHE XUA profile.

6.1 WS-Security (WSS)

The WS-Security specification [WSS] is the basic building block for the web service security which can be used within several security models (PKI, Kerberos, etc) and supports for multiple security token formats, multiple encryption and signature formats and multiple trust domains. The WS-Security specification provides three main mechanisms:

- Message integrity
- Message confidentiality
- Ability to send security tokens as a part of the message (in SOAP headers).

The message security, integrity and confidentiality of the message, are provided by XML-Signature and XML-Encryption in the WSS specification in conjunction with security tokens. These two security functionalities are provided by the ATNA profile (using TLS) in IHE security model. However, there is a difference; TLS provides transport layer security between two nodes and secure the whole communication and hence does not handle intermediaries. On the other hand, WSS has an end-to-end message level security model which provides flexible security while messages traverse through intermediaries. If IHE need such intermediaries, the message level security provided by WS-Security can be used in conjunction with TLS.

The third functionality of the WSS specification is the most important one for the IHE XUA profile in web service transactions since the security tokens are used to carry user identity information with original requests. The WSS specification supports different security token formats which can be classified as:

- User Name Token
- Binary Security Tokens (e.g. X509 certificates, Kerberos tickets)
- XML Security Tokens (e.g. SAML).

The WSS Specification has defined some token profiles which show how the tokens can be transmitted in the WSS Security headers. The XUA profile states that the WS-I Basic Security Profile [WS-I-BasicSecurityProfile] and WS-I SAML Token Profile [WS-I-SAMLTokenProfile] should be used for web service transactions. These profiles put some restrictions over the WSS specification and WSS SAML Token Profile [WSSSAMLTokenProfile] in order to provide interoperability.

The WSS specification just specifies how the security tokens are carried with the SOAP request, does not define how the web service provider can establish trust for these security tokens and ensure the identity of the web service requestor. There are other web service standards published for these purposes.

6.2 WS-Trust

The WS-Trust specification [WS-Trust] provides a trust model for the web service environments and methods for issuing, renewing and validating security tokens. These security tokens are used for authentication, identity federation, attribute federation and non-repudiation for web service transactions.

The trust model is shown in the Figure 18. It is similar to the model given in the SAML profiles. The Security Token Service (STS) makes assertions about the web service requestors that it trusts. The assertions are used by Web Service providers which trust to the STS, like Identity Providers do in SAML Profiles. The arrows show the possible communication ways between the actors to establish the trust. The WS-Trust specification does not force any of these communication flows. However, it specifies the services (issuing, renewing, validating security tokens) that the STS should support in order to construct such trust brokering methods. The WS-Federation specification which is built on the WS-Trust defines such mechanisms.

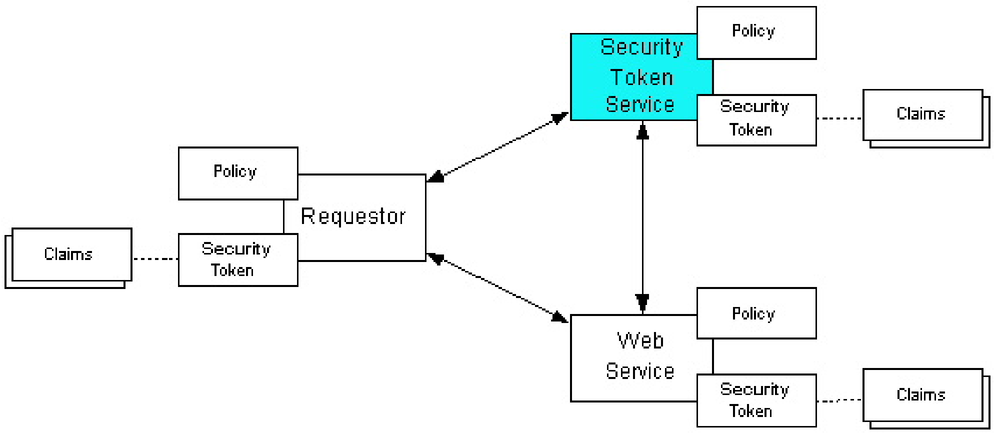


Figure 18 WS-Trust model

The WS-Trust specification defines a generic way for requesting security tokens from the STS. Based on this generic protocol it defines four bindings:

- **Issuance Binding**: Defines the details of the request for a security token for the first time (structure of the request/response message and how to process them).
- **Renewal Binding**: Defines how an existing security token can be renewed if its lifetime has ended in some way.
- **Cancel binding:** Define how a security token issued by the STS can be canceled (all actors which somehow trust the security token should be informed for the cancellation of the security token).
- **Validate Binding:** Define the way how a Web Service can ask the STS for validation of a security token that is previously issued by the STS or somehow trusted by the STS.

The STS may request challenges from the Requestor of the security token. Therefore more transactions may occur between the STS and the Requestor (similar to transactions between IDP and ECP like requesting a fresh authentication). The WS-Trust specification defines negotiation and challenge extensions in order to handle these transactions (e.g. Signature challenge, binary key exchanges, etc).

6.3 WS-SecureConversation

Sometimes the Requestor may need a series of transactions (conversation) with Web service. In these situations, the authentication and other identity information should not be provided for each transaction. The WS-SecureConversation specification [WS-SecureConversation] defines how the web services can establish a security context for a conversation and how this conversation can be secured. In this respect, a special security token, Security Context Token (SCT) is provided in the specification. This token is obtained from the STS by using the WS-Trust specification and used in the conversation between the Requestor and Web Service in conjunction with the WS-Security specification. As mentioned before, the ways of establishing security contexts after getting the assertions from IDPs are not specified in SAML Profiles. The WS-SecureConversation can be used for IHE services to establish such security context for further calls over the service. For example, a user may perform several XDS queries. After the first query, the XDS Registry will have the authentication and other required identity information (assertions) about the user by the execution of the XUA profile. However, if such a security context is not established, the assertions are carried unnecessarily for each XDS Query.

6.4 WS-Federation

The WS-Federation specification [WS-Federation] describes how WS-Security and WS-Trust can be combined in order to construct more complex trust models. It gives several examples which describe several scenarios between different trust domains. The Figure 19 illustrates two given alternative models between two different trust domains. In these models WS-Federation specification combines the Identity Provider and Security Token Service as IP/STS.

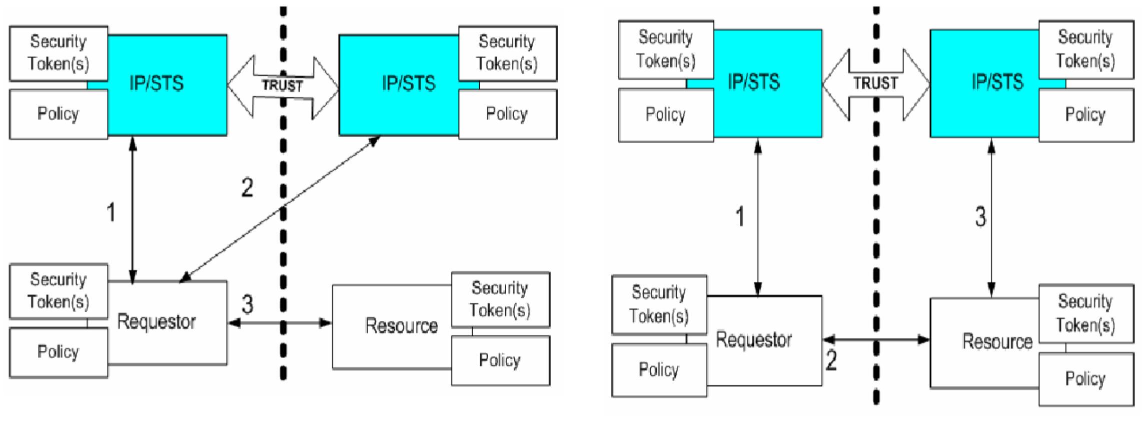


Figure 19 WS-Federation Alternative Trust Models

In the first one, WS-Trust Issuance binding is used two times: 1) to get a security token in order to access IP/STS of the resource, 2) to get a security token in order to access the resource. In the second one, the WS-Trust Validation binding is used in the third step. The Requestor tries to access the resource with a security token obtained from its own IP/STS and the Resource side uses the Validation service of its IP/STS in order to ensure the security token is valid. In both cases, a trust relationship exists between the IP/STS of trust domains.

As shown in the above example, several models can be used for identity federation and cross user authentication in IHE Affinity domains. As we have mentioned in the SAML sections, IHE should specify such models for its defined XUA transactions; Pre-Generated Assertions and Post-Generated Assertions.

In addition, the WS-Federation provides use cases and models for attribute federation and more specifically federation of pseudonyms that is names or identifiers for the users.

6.5 Discussions on WS standards

The above sections briefly describe the main objectives of the WS standards mentioned in the XUA profile (or can be useful). Among these specifications, the WS-Trust specification will be the main element while profiling the cross-enterprise user assertions or authentication for web service transactions. However, simply referencing the current core WS-Trust specification will be not enough to make the XUA an interoperability profile for the specified objective. If we form an analogy between the SAML approach and the WS-Trust specification, the WS-Trust specification corresponds to a part of SAML Core specification which gives protocols (Request/Response) for the communication with the IDP/STS (that is the bindings; Issue, Renewal, Validation). In this respect, the specification itself also mentions the need for additional profiles that will restrict and model the usage of these bindings for several use cases (like SAML profiles do in SAML approach).

While considering such profiles, the first issue is related with the trust model of the affinity domain; 1) Should we have the same model given in the Section 4 that is the each STS has a trust relationship with each web service in the affinity domain or 2) Should we allow having different STSs for a Service Requestor and a web service (like WS-Federation use cases). The second item will be needed desperately for federated affinity domains. Therefore, the profile should handle both of them.

Finding suitable WS-Trust bindings and a communication scenario for the XUA transactions is also very important. The Issuance and Validation bindings may be the basic transactions for the XUA Pre-Generated and Post-Generated Assertion scenarios. In the Pre-Generated Assertion scenario, the Service Requester actor requests the SAML assertion (security token) from the STS by Issuance binding. Then the SAML assertion will be sent to the Web Service within the original web service transaction. However, the web service needs some mechanisms in order to establish trust for the received assertion. Therefore, the STS should bind a Proof of Possession (POP) Token to the security token it sends. The POP will prove that the SAML assertion is generated at the STS. In the other scenario, the requestor will directly initiate the web service call. It can attach its own security token (as credential, may be a SAML Assertion, X509 Certificate of the user, etc) to this request. The web service will use the credential and request Validation from the STS. The STS can return the SAML assertion related with the user as a response to the request.

After deciding the way of communication for assertions and the WS-Trust bindings used in these communication flows, the issues related with the content of the SAML assertion should be resolved.

7 BPPC and XACML

In this section we continue giving examples from our implementation. We describe how we use Extensible Access Control Markup Language (XACML) [XACML2.0] and its RBAC profile [XACML-RBAC] to express consent policy and to implement the authorization service for the XDS Retrieve Document transaction. In addition, we discuss on IHE Basic Patient Privacy Consent (BPPC) [IHE-BPPC] profile and how XACML can be used within the profile.

7.1 XACML Model

XACML is an XML based mark-up language for the policy management and access decisions. XACML standard not only gives the model of the policy language, but also proposes a processing environment model to manage the policies and conclude the access decisions. In addition, it defines the Request/Response protocols for the communication between the application environment and the policy decision environment.

XACML represents the access control rules based on four main structures: *Subject*, *Resource*, *Action*, and *Environment.* Basic data source for defining the policy and the *Request* and *Response* messages are the attributes related with these main structures. For example, variety of the *Subject* attributes like name of the subject, email of the subject, role of the subject (both functional and structural), etc can be used in policy decisions. The *Resource* attributes may be resource ID or classification of the resource in terms of some criteria like sensitivity or type of the document. In addition, other attributes may be used like owner of the document (e.g. patient), time it is created or submitted to the system, the author institution or the author itself if we consider the Healthcare domain. The *Action* attributes are generally single identifiers like ‘read’, ‘write’, ‘update’ or ‘delete’. However, the action can also be defined with more than one attributes in more complex situations. The *Environment* attributes defines the current environment which is independent of other structures (e.g. current time).

Another source of data to use in XACML policy decisions and transactions is the requested resource itself. XACML use XPath standard for the XML based resources. For example, an EHR document may include demographic data of the patient and the access policy may have a statement about the age of the patient. In this case, the age information may be retrieved by XPath expressions given in the policy definitions. Furthermore, in this way, not the whole resource but the parts which are permitted by the XACML policy can be provided to the requester. However, in order to perform such operations, the structure of the resource must be fixed, it must be XML and it must be known by the authors of the XACML policy. For the healthcare domain, this can be achieved only for strictly regulated and specialized systems in terms of content and communication.

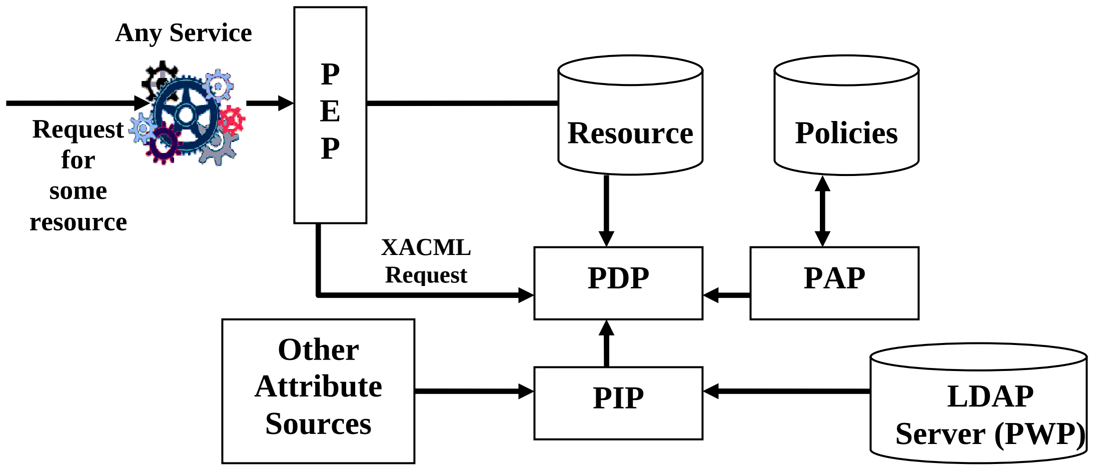


Figure 20 XACML Processing Environment

The XACML specification also defines a processing environment model as illustrated in the Figure 20. Four actors are mentioned in this processing environment; Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Administration Point (PAP) and Policy Information Point (PIP).

PEP is the entry point for the access control mechanism which isolates the XACML processing environment from the application environment. It requests the access decision from the PDP by sending an XACML *Request* message. The *Request* message includes the attributes about the resource, subject and action as described above. These attribute values are obtained from the service which requests the resource. SAML technology and assertions are the best example for cross enterprise services which shows how such information can be obtained. However, providing all attributes in the *Request* message is not dynamic since the PEP should know all the required attributes for the execution of policy before asking for the decision. XACML specification proposes the PIP entity for this purpose. When the PDP needs value of an attribute it asks the PIP which knows how to obtain the values from outside services or from attribute values exist in the XACML *Request*. In our implementation, we simply assume that the required attributes for the consent policies are invariable and known by the PEP. Therefore, the PEP obtains the attribute information by using the SAML Assertion Query/Request profile from the IDP and provides the attribute values within the XACML *Request*. As seen from the Figure 20 and above discussions, several possible alternative access control architectures can be provided by using the XACML and SAML (most suitable for XACML) standards. In the following sections we provide more discussions about how XACML can be used for IHE BPPC and further consent or privacy policy related profiles.

7.1.1 Example XACML policies

Today access control models are mostly based on RBAC model. XACML also has published an RBAC profile which defines how the XACML can be used to construct RBAC policies. This profile divides the policies into two types; *Role* policies and *Permission* policies. The Figure 21 and the Figure 22 shows the example *Role* and *Permission* policies for the XACML RBAC profile.

The Figure 21 illustrates a *Permission PolicySet* which defines the rules to access the medical records which are annotated by the ‘GeneralClinicalInformation’ value as their sensitivity level. As seen from the example, the sensitivity value of the resource is obtained from the *urn:oasis:names:tc:xacml:1.0:resource:confCode* attribute which is a resource attribute in the XACML Request message coming to the PDP.

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" PolicySetId="PPS:MEDICALDOCTOR:Role"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides">
 <Target/>
 <Policy PolicyId="Policy1" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-
overrides">
  <PolicyDefaults><XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-19991116</XPathVersion></PolicyDefaults>
  <Target>
   <Resources>
        <Resource>
         <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">GENERALCLINICALINFORMATION</AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:confCode"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
         </ResourceMatch>
        </Resource>
   </Resources>
  </Target>
  <Rule RuleId="Role:ConfCode:Rule1" Effect="Permit">
   <Target/>
   <Condition>
        <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-range">
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
DataType="http://www.w3.org/2001/XMLSchema#time"/>
         </Apply>
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">08:00:00+02:00</AttributeValue>
         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">20:00:00+02:00</AttributeValue>
        </Apply>
   </Condition>
  </Rule>
  <Obligations>
   <Obligation ObligationId="tr:edu:metu:srdc:xds:pep:obligations:mail" FulfillOn="Permit">
        <AttributeAssignment AttributeId="tr:edu:metu:srdc:xds:attribute:mailto"
DataType="http://www.w3.org/2001/XMLSchema#string">
         <ResourceAttributeDesignator AttributeId="tr:edu:metu:srdc:xds:attribute:patient-email"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </AttributeAssignment>
        <!--Other required attributes for mail obligation-->
   </Obligation>
  </Obligations>
 </Policy>
</PolicySet>
```

Figure 21 Permission PolicySet

The XACML *Condition* element defines the rules to access the specified type of resource. In our example, it states that GeneralClinicalInformation are accessible only between hours 08:00 and 20:00. The *Obligation* element states that if the decision is permited for this Permission Policy then the obligation with the identifier *tr:edu:metu:srdc:xds:pep:obligations:mail* must be realized. The *Obligation* concept provides some functionality to the system and policy makers to define responsibilities and obligations for the system or the requester of the resource which must be obeyed after the decision is taken. For example, the obligation that is shown in the Figure 21 forces the system to inform the patient with an email after the resource access is granted. The attributes that will be used to perform the obligation are also included in the obligation (e.g. the mail address of the patient). Obligations are processed by the PEP.

The *Role PolicySet* is illustrated in the Figure 22. It requires matching the *urn:oasis:names:tc:xacml:1.0:subject:role* attribute to the value ‘MedicalDoctor’ for the applicability of the PolicySet. The *PolicySetIdReference* elements give references to the Permission PolicySets which are processed further to decide on the access decision for the subject with the ‘MedicalDoctor’ role.

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
PolicySetId="RPS:MEDICALDOCTOR:Role" PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-
combining-algorithm:permit-overrides">
  <Target>
    <Subjects>
      <Subject>
              <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">MEDICALDOCTOR</AttributeValue>
                <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
              </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <PolicySetIdReference>PPS:MEDICALDOCTOR:Role</PolicySetIdReference>
</PolicySet>
```

Figure 22 Role PolicySet

7.2 BPPC Discussions

Our scenario has some differences with the IHE BPPC Profile in terms of consent management model. In our scenario as described before, patients create their privacy consent policies which are then sent to XDS Repository. Then, when a document is requested, the privacy consent policy which the patient gives to the institute is found by the XDS Repository and used for the access control decision. This scenario is not practical in real life in terms of legal issues (An institute may not be able to get consent for the future documents. As another case, a consent policy is needed to be created by the patient for a specific document or a group of document which may be a tedious case). However, it may be beneficial in some special cases where patient has to define some access control rules for specific health records as mentioned in one of the BPPC use case; ‘Policies in an environment with comprehensive access controls’. On the other hand, in the normal use cases of the BPPC profile, the Privacy Consent Policies are provided by the policy makers of the Affinity Domain. The patients select some of these policies and sign a consent document that references to these consented policies. The healthcare institutes in the affinity domain must obey the rules in the consented policies.

The enforcement point for the access decision is located at the client side (Document Consumers) in BPPC profile. In our implementation the access control enforcement is performed at the service side (XDS Repository). In this respect, the BPPC profile assumes a strong trust on the document consumer systems for applying the Privacy Consent Policies. With this assumption, the system will become more simple since the information (attributes, user identification, etc) for access control mechanisms that is located at the client side does not need to be transferred to the service side. Nevertheless, this type of information is needed for auditing. In fact, this assumption can not be accepted by some other business domains in terms of security and privacy requirements. However, when we consider the situation together with the IHE affinity domain concept and since the enforcement is related with patient consents, the trust assumption seems reasonable and practicable. On the other hand, the Service Providers (XDS Repositories, XDS Registries, PIX Managers, etc) in the affinity domain should have their own privacy policies and access control mechanisms which may require the same user information from the client side.

BPPC does not restrict the content of the Privacy Consent Policies. Therefore, the implementations of the access control mechanisms will be manual and specific to the Privacy Consents defined in the Affinity Domain. On the other hand, the implementation of the mechanisms will be very easy if IHE selects a machine processable access policy standard for the Privacy Consent Policies. The XACML standard seems to be very suitable for this purpose. It can provide all functionalities for the access control systems mentioned in the BPPC profile. In addition, it also supports more complex functionalities for future refinements and the profiles. The following section discusses the use of the XACML to represent the Privacy Consent Policies of the affinity domain mentioned in the BPPC profile.

7.2.1 Using XACML for Privacy Consent Policies

The BPPC Profile provides a possible implementation way of Privacy Consent Policies. It uses an access control matrix consisting of roles and sensitivity markers. The matrix can be sliced in several ways to form the Privacy Consent Policies. The XACML language can be used in any way that is used to divide the access control matrix.

The first example is using policies which describe the whole access control matrix. In this way several access control matrices are generated and each of them is represented by single Privacy Consent Policy describing the preferences about all the sensitivity markers. The patient should select only one of them since the policies describe different matrixes and they may be incompatible. Using such a methodology may not be preferred since the description of the policy is difficult and complex to make the patients understand them. If such a methodology is used, the Privacy Consent Policies should be named or classified in terms of their restrictive capabilities (e.g. loose, restrictive, very restrictive, etc).

If the matrix is divided based on either the role vocabulary or sensitivity markers, the methodology defined in the XACML RBAC profile can be used. In this case, the Privacy Consent Policies put restrictions for a role defining which sensitivity markers the role is allowed to access or for a sensitivity marker defining which roles are allowed to access the resource for the defined sensitivity marker. We will give the example for the sensitivity marker based slicing since naming the Privacy Consent Policies with the sensitivity markers is more suitable (it is more meaningful to publish resources with the confidentiality code corresponding to sensitivity markers). As we show in the examples, we divide the policies into two types. However, we give references from Permission policies to Role policies that is Permission Policies will be executed first. Other restrictions (e.g. time) can be put on the Role policy.

```
<PolicySet                    PolicySetId="urn:ihe:bppc:privacyconsentpolicies:example:SensitiveInformation"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable ">
  <Target>
    <Resources>
      <Resource>
            <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue>SensitiveInformation</AttributeValue>
              <ResourceAttributeDesignator AttributeId="..."/>
            </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <!-- Obligations for the Privacy Consent policy in case of the decision is permit or deny-->
  <PolicyIdReference>
      urn:ihe:bppc:rolepolicies:example:DirectCareProvider
  </PolicyIdReference>
  <PolicyIdReference>
     urn:ihe:bppc:rolepolicies:example:EmergencyCareProvider
  </PolicyIdReference>
</PolicySet>
```

Figure 23 Privacy Consent Policy defined by XACML

The Figure 23 illustrates a Privacy Consent Policy for an Affinity Domain defined by XACML. The *PolicySetId* can be used as a unique Privacy Consent Policy identifier. The XACML *Target* element gives the information in order to decide if the *PolicySets* or Policies are applicable or not. In our example, if the resource is not classified as 'SensitiveInformation' then this Privacy Consent Policy is not applicable. If the Privacy Consent Policy is applicable, the role policies given by references will be executed. The policy combining algorithm defines the way of execution and combination of the results. The *Only-One-Applicable* algorithm states that only one policy can be selected as applicable and overall result is the result of the applicable policy. The Figure 24 shows the *Role* policy which is referenced from the Privacy Consent Policy. If the target matches then the rules (other restrictions) will be evaluated and result is returned. Any Rule combining algorithm can be selected according to the rules and preferences defined in the policy. If there are no restrictions, a single rule stating the rule effect as Permit will be enough. The corresponding obligations in Role Policies and the general obligations defined in Privacy Consent Policy should be executed after taking the decision.

```
<Policy PolicyId="urn:ihe:bppc:rolepolicies:example:DirectCareProvider"
RuleCombiningAlgId="Depends-On-Your-Choice-Over-Rules">
  <Target>
    <Subjects>
      <Subject>
            <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
              <AttributeValue>DirectCareProvider</AttributeValue>
              <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:role"/>
            </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <!-- Rules: other restrictions-->
  <!-- Obligations-->
</Policy>
```

Figure 24 Role Policy for BPPC

Other methodologies can also be produced by defining different policy combining algorithms. Currently, XACML has provided six policy or rule combining algorithms. The main algorithms are Permit-Overrides, Deny-Overrides, First-Applicable, Only-One-Applicable. The XACML RBAC uses permit overrides to combine the permission policies. In the above example, we use Only-One-Applicable algorithm since we assume a user can have only one role. The discussion can also be applied to Privacy Consent Policies. If it is assumed to have a document with more than one sensitivity marker, then an appropriate policy combining algorithm must be chosen to combine the Privacy Consent Policies. However, BPPC profile does not allow multiple roles or multiple sensitivity markers.

The Document Consumer actors which implements BPPC profile with such Privacy Consent Policies in XACML format should execute the following steps:

1) Find the Privacy Consent Policies (XACML PolicySets) which the patient has given consent.
2) Combine them to single PolicySet with Only-one-applicable algorithm
3) Determine the access grant;
    a) If the PolicySet evaluates to Permit, grant access
    b) If the PolicySet evaluates to NotApplicable, deny access
    c) If the PolicySet evaluates to Deny, deny access

The XACML evaluation models are based on matching of the conditions in the rules and targets in policy and policy sets. The policy maker can only state ‘deny’ or ‘permit’ decision while giving the effect of a rule. For example in the above example, if an administrator tries to access a record classified as ‘SensitiveInformation’, then the target in the PolicySet shown in Figure 23 will match. However, the two role policies will not match with the ‘Administrator’ role. In this case, the PolicySet will evaluate to ‘NotApplicable’, not to ‘Deny’. To produce the ‘Deny’ result, the PolicySet should include the ‘Administrator’ role policy and this policy has a rule with effect value ‘Deny’. Such rules are called negative-rules. However, negative rules are not recommended by authorities since they can lead to policy violations. The XACML support negative rules but it recommends not to use them. The BPPC profile also mentions negative rules and states that they must not be used. Therefore, the step 3.c will never occur if negative rules are not used and the ‘Not Applicable’ result (3.b) implicitly defines a deny situation.

References

[IHE-ATNA] IHE Audit Trail and Node Authentication Profile,
http://www.ihe.net/Technical_Framework/upload/ihe_iti_tf_2.0_vol1_FT_2005-08-15.pdf

[IHE-BPPC] IHE Basic Patient privacy Consent Profile,
http://www.ihe.net/Technical_Framework/upload/IHE_PCC_TF_BPPC_Basic_Patient_Privacy_Consents_20060810.pdf

[IHERoadmap] IHE IT Infrastructure Planning Roadmap,
http://www.himss.org/Content/files/IHE_IT_Roadmap_Final2005.pdf

[openSAML] An Open Source Security Assertion Markup Language implementation,
http://www.opensaml.org/

[SAML2.0] OASIS Security Assertion Markup Language Core Specification, http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[SAMLAuthContext] OASIS Security Assertion Markup Language Authentication Context Specification, http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf

[SAMLBindings] OASIS Security Assertion Markup Language Bindings Specification,
http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf

[SAMLMetadata] OASIS Security Assertion Markup Language Metadata Specification,
http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf

[SAMLProfiles] OASIS Security Assertion Markup Language Profiles, http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

[SAMLTechnicalOverview] OASIS Security Assertion Markup Language Technical Overview,
http://www.oasis-open.org/committees/download.php/20645/sstc-saml-tech-overview-2%200-draft-10.pdf

[SAML-ImplGuideline] OASIS Security Assertion Markup Language Implementation Guidelines, http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip

[SSL3.0] SSL 3.0 Specification, http://wp.netscape.com/eng/ssl3/

[TLS1.0] TLS Protocol v1, http://www.ietf.org/rfc/rfc2246.txt

[WS-Federation] Web Service Federation Language, ftp://www6.software.ibm.com/software/developer/library/ws-fed.pdf

[WS-I-BasicSecurityProfile] WS-I Basic Security Profile v1, http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html

[WS-I-SAMLTokenProfile] WS-I SAML Token Profile v1, http://www.ws-i.org/Profiles/SAMLTokenProfile-1.0.html

[WSS] WS Security Core Specification v1.1, http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[WSS-SAMLTokenProfile] WSS SAML Token Profile v1.1, http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf

[WS-SecureConversation] OASIS WS-SecureConversation v1.3, http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-rddl.html

[WS-Trust] OASIS WS-Trust v1.3 , http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-rddl.html

[XUA2006] IHE Cross Enterprise User Authentication (XUA) 2006-2007 White Paper, http://www.ihe.net/Technical_Framework/upload/IHE_ITI_TF_White_Paper_CrossEnt_User_Authentication_PC_2006-08-30.pdf

[XACML2.0] OASIS Extensible Access Control Markup Language Core Specification, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
[XACML-RBAC] OASIS Extensible Access Control Markup Language Core and hierarchical role based access control (RBAC) profile http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf