Web Services Trust and XML Security Standards

Date: April 9, 2001
Version: 1.0

Entrust®
Securing Digital Identities & Information

Introduction

Web services are self-contained, modular applications that can be described, published, located, and invoked over the Internet. They perform well-defined functions both for applications and other Web services, which can be anything from simple calculations to complicated business processes. Through their loose-coupling and dynamic real-time discovery and binding, Web Services insulate applications from the complexity and details of other components, creating systems that are more flexible and adaptable.

The fact that Web services will use the inherently insecure Web for possibly mission critical business transactions, and the possibility of new short-lived and dynamic business relationships that Web services enable, mean that integrating trust into Web services poses challenges for a Web services Trust architecture. Any such security architecture must address issues of:

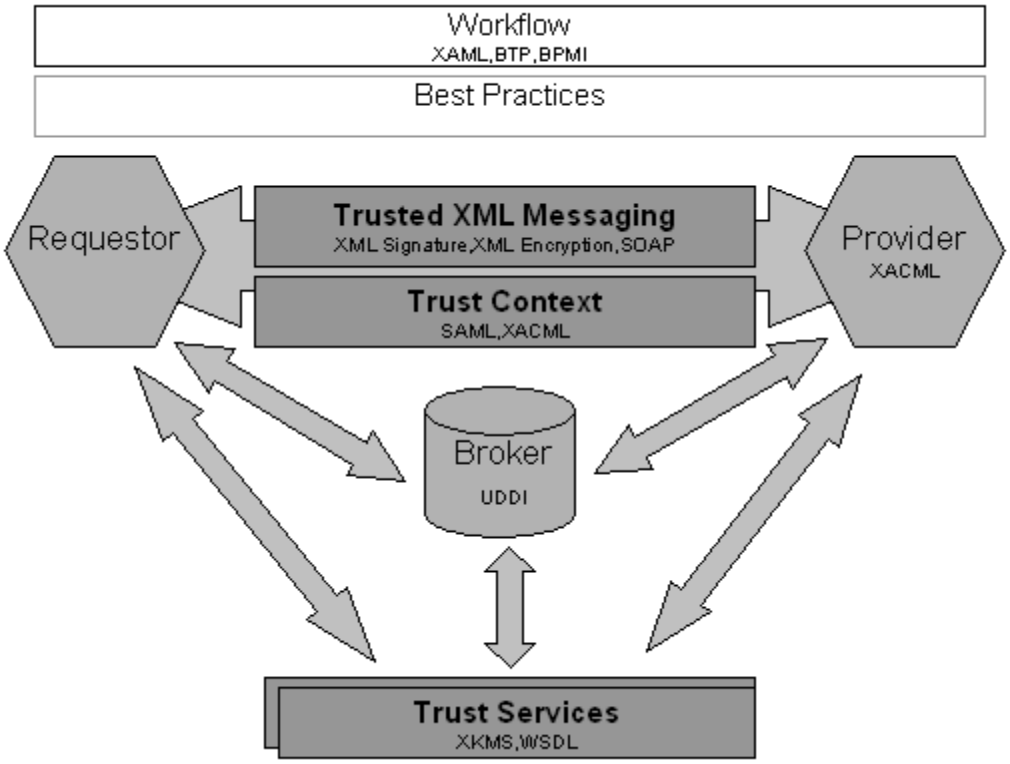
- Authentication — how can the service provider be confidant that the requestor is really who they claim to be?
- Authorization — how can the provider determine whether the requestor is authorized to use the service?
- Confidentiality — how can the business transaction data be protected from unauthorized access as it is transmitted between requestor and provider?

- Integrity — how can the service requestor and provider be confident that the business transaction data was not tampered with in transit?

- Non-repudiation — how can the service requestor and provider be confident that the other entity cannot deny their participation in the transaction?

The Web Services Trust Framework from Entrust provides a framework for describing the components of a general Web Services trust architecture; distinguishing between **Trusted Messaging**, **Trust Services**, and **Trust Context**. It is Trusted Messaging, securing the communications between service requestor and service provider that comprise a transaction, that is most fundamental. Trust Services and Trust Context provide mechanisms and processes in support of Trusted Messaging.

The Web Services Trust Framework is shown in the diagram below, along with the XML standards back plane on which it is built.



Here the three fundamental Web Services conceptual components are shown: **service providers**, who publish descriptions of and invocation details for the services they provide to a **service broker** such that **service requestors** can search for these services and retrieve the detailed information necessary to bind to them.

Shown as an overlay are the components of the Web Services Trust Framework. The requestor and provider interact through the exchange of suitably encrypted and signed **Trusted Messages**, possibly accompanied by additional trust information necessary to establish the identity and entitlements **Trust Context** of each participant. Additionally, either the requestor or the provider

(or both) may need the services of a **Trust Service**, for which they search the broker registry, and bind to like any other Web Service.

The following sections describe in more detail these components, the relevant standards and how they all work together within the Web Services Trust Framework.

Trusted Messaging Standards

The fundamental interaction in Web services is the transmission of messages between the service requestor and the service provider. (Note: this is not to suggest that there are not important security considerations for the service broker, e.g. ensuring that only authenticated service providers can modify the appropriate business and service descriptions. However, since the broker will itself be a service provider — providing business and service description publication and searching functionality to other service requestors — the transactions with the broker are still that of requestor-provider.)

While the different Web services platforms typically support other protocols and syntaxes, the default is that these messages will be XML — most likely contained in a SOAP envelope — with HTTP as the transport protocol. In some sense then, the fundamental challenge for enabling Trusted Web Services is ensuring that these base XML messages, on which the Web services model is built, can be trusted. For an XML message, trusted means that its origin can be authenticated, that its confidentiality can be assured, that its data integrity can be determined, and that the participants cannot later deny either sending it or receiving it.

The following sections discuss the standards and/or proposed standards that will together enable Trusted Messaging.

XML Signature

Data integrity and non-repudiation are critical for B2B transactions. This fact was recognized when, on June 16, 2000, the U.S. Senate approved by a unanimous vote the “Electronic Signatures in Global and National Commerce Act” (the E-Sign Act). The E-Sign Act is landmark legislation because it gives the same legal recognition to electronic signatures as has existed for written signatures. No longer will these electronic methods be open to question. Billions of dollars of business-to-business and business-to-consumer transactions will be facilitated as written signatures and paper bills, receipts etc. will no longer be required.

The joint IETF-W3C XML Signature working group was tasked with developing an XML syntax for representing signatures on digital content along with procedures for computing and verifying such signatures. Whilst the signatures themselves are captured in an XML syntax, XML Signature is not limited to signing only XML documents.

A fundamental feature of XML Signature is the ability to sign only specific portions of the XML tree rather than the complete document. This will be relevant when a single XML document may have a long history in which the different components are ‘authored’ at different times by different parties, each signing only those elements relevant to itself.

This flexibility will also be important in situations where it will be important to ensure the integrity of certain portions of an XML document, while leaving open the possibility for other portions of the document to change, e.g. a signed XML form delivered to a user for completion. If the signature were over the full XML form, any change by the user to the default form values would invalidate the original signature.

There are in general three alternatives for the location of the XML Signature relative to the source XML: these are enveloped (where the XML fragment that makes up the signature is inserted into the source XML), enveloping (where the XML signature wraps the source XML), and external (where the XML Signature is located in a separate XML document to the source). The listing below gives a sample XML Signature for the external case. Here the signature is computed over the Entrust Web site home page, as identified by the URI attribute of the <Reference> element. The signature itself is stored in the <SignatureValue> element. Somewhat interesting is to note that both changes to the content of the specified HTML page, as well as the unavailability of the page, would invalidate the signature.

```
<?xml version="1.0"?>
<Signature Id="MyFirstSignature" xmlns=http://www.w3.org/2000/09/xmldsig#>
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026"/>
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
<Reference URI="http://www.entrust.com/index.html">
<Transforms>
<Transform Algorithm="http://www.w3.org/TR/2000/CR-xml-c14n-20001026"/>
</Transforms>
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
</Signature>
```

The signature is computed for the Entrust Web home page.

An important issue for signing is the fact that an XML document will often not exist in isolation; it may often be accompanied by schemas and perhaps associated style sheets. Since it may be the case that these associated components contribute meaningfully to the ultimate information content of the XML document, an assertion about the XML document (made explicit through the signature) may need to include these 'secondary' objects.

Additionally, since an XML document will typically need to be transformed or formatted before presentation to a user, having the user sign the XML itself, and not the HTML from which they actually viewed the data, may not provide non-repudiation of the transaction for which the user's digital signature is required. It may be necessary to have the user sign all the components (HTML, stylesheets, graphics, etc) that combine to form the interface actually viewed. Otherwise,

the user could later deny the transaction by claiming that certain critical information wasn't presented in an appropriate manner, e.g. the total transaction cost was in an unintelligible font.

The XML Signature proposal, of which Entrust is a co-author, is currently a W3C Candidate Recommendation under a joint W3C/IETF Working Group.

Canonical XML

To test the integrity of a signed XML document, i.e. verify that it has not been tampered with, the recipient must compare the digest of the document to the decrypted digest that formed the signature. If the two digests match, then the integrity of the document can be trusted. The digest calculation is sensitive to the slightest modification of the XML document, even those that may not change the information content, e.g. an extra space between an element name and the closing tag delimiter '>'. Unfortunately, different XML applications, while uniformly processing the information content, may treat the physical representation of the XML document differently. Since it may well be the case that the XML document is processed by interim XML applications between the time that the document is signed and the signature is verified, it is possible that the digest comparison could fail, resulting in the signature not being trusted.

Canonical XML addresses this concern; the *canonical* form of an XML document is a normalized physical representation that establishes a standard baseline for signature processing. During signature generation, the digest is computed over the canonical form of the document. The document is then transferred to the relying party, which validates the signature by reading the document and computing a digest of the canonical form of the received document. The equivalence of the digests computed by the signing and relying parties (and hence the equivalence of the canonical forms over which they were computed) ensures that the information content of the document has not been altered since it was signed.

The XML Signature <Transforms> element is used to communicate to the Relying Party what transformations were applied to the complete document before the signature was calculated. The Relying Party then applies the same transformation before calculating the digest. The <transforms> element can contain multiple <Transform> elements. The output of each Transform serves as input to the next Transform. The following XML fragment displays the <transforms> element for a signature in which only the <salary> element of the source XML was signed, after which it was canonicalized.

```
<Transforms>
 <Transform>
  <XSLT>
  <xsl:stylesheet>
  <xsl:template match="salary">
   <xsl:copy-of select="."/>
   </xsl:template>
   </xsl:stylesheet>
   </XSLT>
  </Transform>
  <Transform Algorithm="http://www.w3.org/2000/CR-XML-c14n-20001026"/>
</Transforms>
```

This <Transform> element specifies that only the <salary> element is to be signed.

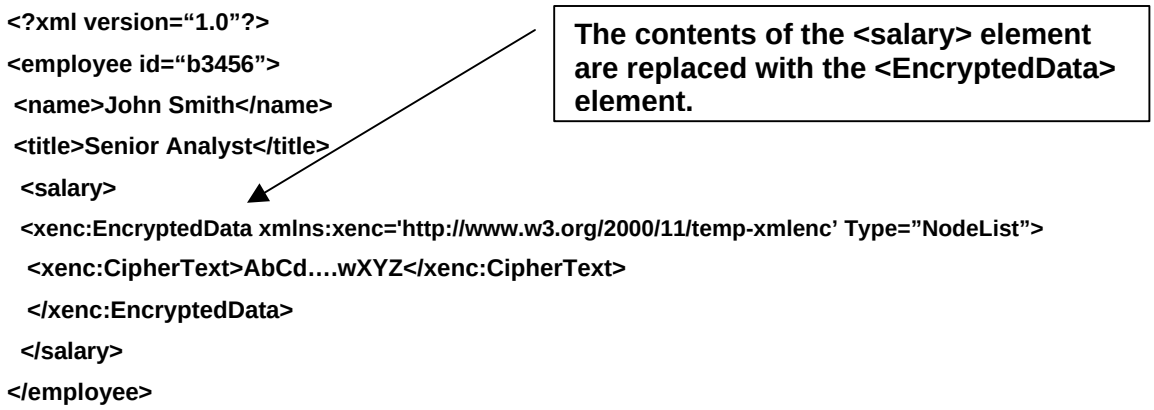
This <Transform> element indicates the canonicalization process.

Canonical XML has reached the W3C Candidate Recommendation stage.

XML Encryption

When transmitting an XML document over the Internet, protocols like SSL (TLS) or S/MIME either secure the transport pipe or the complete XML document to protect the confidentiality of the XML message. There will, however, be situations in which only certain parts of an XML document need be (or even should be) encrypted, leaving the rest in plain text. In general, these situations are those in which the requirements of data confidentiality must be balanced against the requirements for data processing by parties other than that for which the XML document was encrypted.

For instance, the following listing is a sample of a Human Resources employee record. To ensure that only appropriate personnel can view the <salary> element, its contents have been encrypted and replaced with the <EncryptedData> element. Because the other information objects remain in plain text, they would be available for other processing, e.g. determining how many employees have the 'Senior Analyst' title.



Such 'element-wise' processing is a key requirement of any proposal for encrypting XML. Entrust is a co-author of the XML Encryption proposal which has been submitted to a W3C Working Group for standardization. Similar to XML Signature, XML Encryption specifies an XML format for capturing an encrypted data, but is not limited to dealing only with XML as input.

Because the schema to which an XML document conforms will typically be available (either directly or manually retrievable) and since element names can be quite large, this knowledge could theoretically give an advantage to a known plain-text attack. Consequently, whatever algorithm is used for XML encryption, it should be specifically resistant to known plain-text attacks. The better-known encryption algorithms are fairly resistant to such attacks, and it should not be a problem if recognized standards are followed.

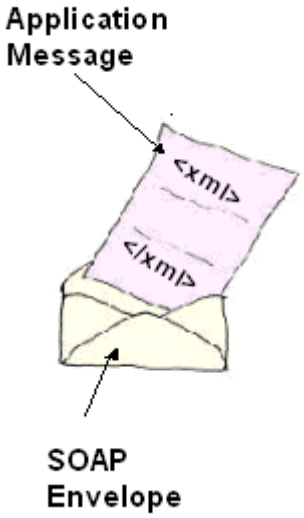
The very nature of encryption, that of obfuscating some portion of data, has serious implications for the schema validity of the resultant cipher XML. For example, if the schema for the Human Resources 'employee' record shown above dictates that the <salary> element contains textual data of a certain type, it is apparent that the encrypted resultant XML will not satisfy this requirement, and that the resultant cipher XML, while 'well-formed', will not be 'valid'. This

may not be an issue for some applications. However, for those scenarios where the encrypted XML document must be processed by a 3rd party intermediary for reasons unrelated to the secure content, this lack of schema-validity may be problematic. Motivated by the application bloat from error-catching code, one of the key principles on which XML was designed was that a parser, on confronting an error in an XML document, should discontinue the parsing process rather than try to proceed. There are a number of possible mechanisms by which this issue could be addressed, ranging from an accompanying transform of the schema itself to ensure validity to introducing the concept of partial-validity.

The W3C XML Encryption Work Group is completing its first Working Draft based on a proposal that Entrust co-authored.

SOAP

SOAP (Simple Object Access Protocol) is a lightweight protocol for XML messaging in a distributed environment. SOAP consists of three things: an envelope for containing XML messages, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.

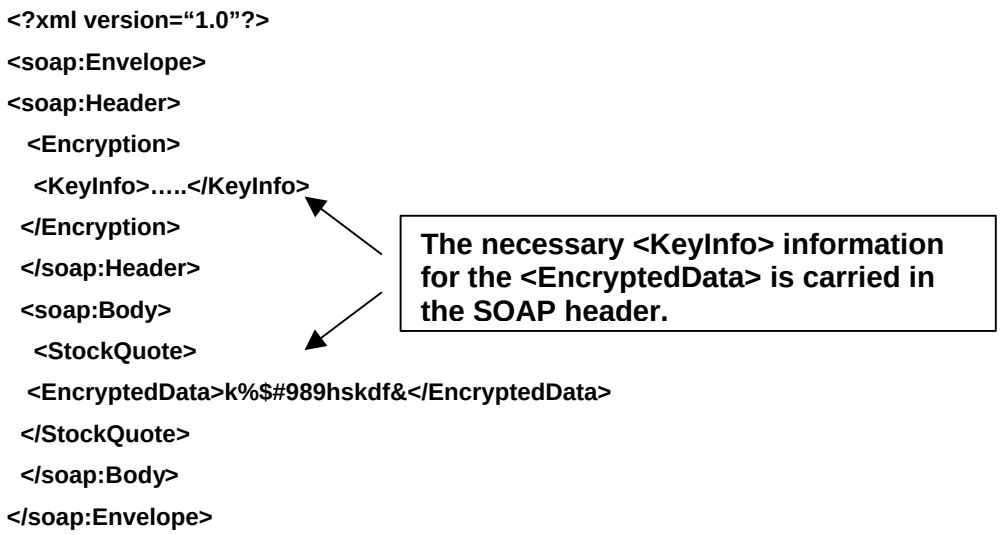


The SOAP Envelope consists of two parts: Header and Body. The Header is a generic mechanism for adding features to a SOAP message. The Body is a container for the XML application data that comprises the real business message.

SOAP itself doesn't address security issues, relying instead on transport-layer security, e.g. SSL, to provide authentication, integrity and privacy. Transport layer security however, is insufficient if end-to-end security is required, e.g. supporting a scenario where an XML document travels over HTTP for some steps of a multiple-hop journey and SMTP for the rest. Such a scenario implies that security information would need to be mapped from one transport protocol to the next at each intermediate stage, opening up the possibility that the plaintext data would be temporarily made accessible in the interval.

Rather than relying on transport layer mechanisms, there are proposals for adding security directly to SOAP messages through the inclusion of XML Signature and Encryption tags in

the header of the SOAP message. Here it is the messages themselves that are secured, enabling both end-to-end security and intermediate processing of non-sensitive data. In the diagram below, the SOAP message for a Stock Quote service is shown in stylized form. The stock ticker symbol has been replaced with the EncryptedData element. All the additional information that would be required to decrypt this cipher text has been moved into the SOAP header, where it could be referenced as required.



Trust Services Standards

Application developers have traditionally been forced to become more proficient in security concepts and technologies than they might desire in order to integrate trust into their applications. Not too surprisingly, they would rather spend their energy on the application development itself. Consequently, trust integration has been perceived as both complex and difficult. Additionally, the typical mechanism by which trust is implemented is through vendor PKI toolkits, resulting in a relatively tight coupling between application and a particular PKI vendor and possibly resulting in interoperability issues for functions like digital certificate processing and revocation status checking. By default, a Web services developer will face these same issues if they wish to integrate trust into that Web service. However, these issues, complexity of implementation, a desire for looser couplings, and flexibility in the choice of a trust provider, are of course the very same that motivate Web services themselves. We should therefore not be surprised then to see that the Web services model can simplify how trust can be integrated into other Web services.

A Trust Service is a Web service that other Web services will invoke to enable trust for their own transactions. Trust Services will deliver the security functions, e.g. signing, encryption, time-stamping, and the accompanying administrative functions, e.g. key registration, revocation, validation, that other Web services will require if they are to guarantee the trust of their transactions. Rather than be confronted with the full complexity of a PKI, the Web service developer now need only be able to invoke the appropriate Trust Service at the appropriate time. For instance, if it is necessary for a Service Provider to calculate a digital signature for the XML service response, the calculation can be outsourced to a Digital Signature Trust Service. The

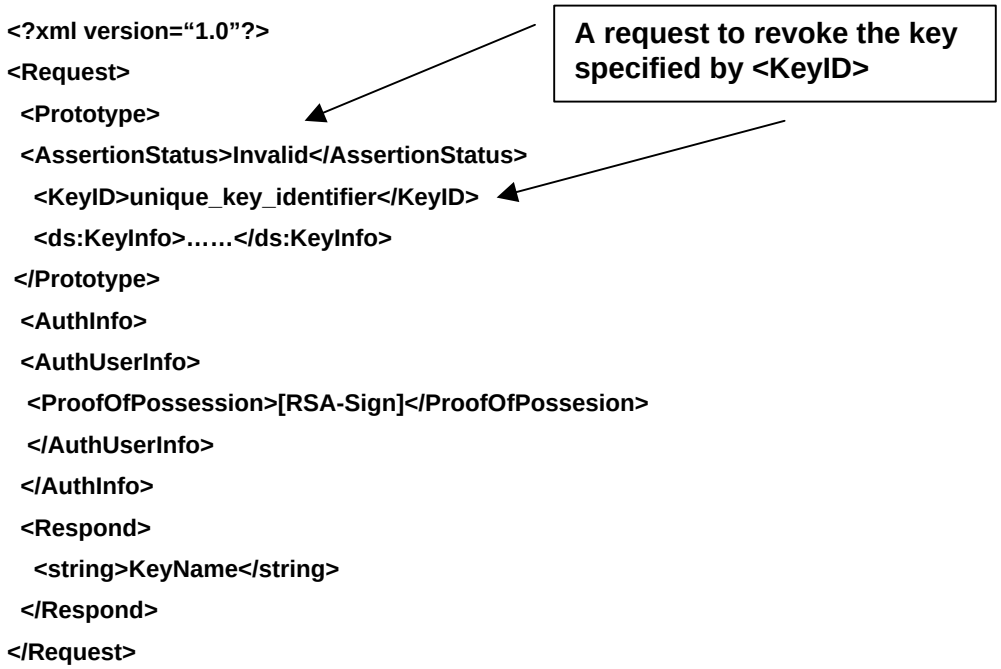
SOAP request for a digital signature would include both the XML source as well perhaps as a specification of which elements within should be signed.

The first example of a standard for such a Trust Service is XKMS.

XKMS

The XKMS (XML Key Management Specification) provides a standard XML-based messaging protocol by which application developers can *outsource* the processing of key management (registration, verification etc) to trust services accessed through the Internet.

With XKMS, the PKI functionality that would traditionally be tightly integrated into the Web service code resides in remote servers that are accessed through easily programmed XML interfaces. For instance, the following listing is an XKMS message requesting that a given key be revoked.




```
<?xml version="1.0"?>
<Request>
 <Prototype>
 <AssertionStatus>Invalid</AssertionStatus>
  <KeyID>unique_key_identifier</KeyID>
  <ds:KeyInfo>……</ds:KeyInfo>
</Prototype>
 <AuthInfo>
 <AuthUserInfo>
  <ProofOfPossession>[RSA-Sign]</ProofOfPossesion>
 </AuthUserInfo>
 </AuthInfo>
 <Respond>
  <string>KeyName</string>
 </Respond>
</Request>
```

As part of its commitment to XML standards, Entrust will support the XKMS initiative when the W3C accepts the current submission. To reflect this commitment, Entrust has developed and made available (online at the Entrust 'Web Services Trust' Web page) an XKMS Service Reference Implementation with which developers (and non-developers) can experiment to familiarize themselves with the XKMS concepts and XML syntax. Additionally, the existing Entrust/DeviceConnector product for connecting Smart Card Management Systems to an Entrust PKI uses an XML interface very similar to XKMS (in functionality if not XML Schema). Entrust is committed to ensuring that the particular requirements of the manufacturing scenario, e.g. batch registration requests, be reflected in future versions of XKMS and will work within the appropriate standards body to make this happen.

Trust Context Standards

While the fundamental Web service interaction may be the communication of application specific business messages between a requestor and provider, e.g. a <requestForQuote>, this will typically be insufficient to allow the two participants to enter into the transaction with an acceptable level of confidence. *Trust context* is the additional information that must be present in order to support a trusted business transaction, e.g. verification of creditworthiness. It is this context that allows participants to make informed decisions about the trustworthiness of other participants and the transaction as a whole. In the past this context would have been established based on a past business history, an assertion of confidence from a trusted 3rd part like a bank, or even personal relationships. Since, in the Web services model these long-term relationships will be (at least potentially) replaced by dynamically built, short-lived and even disposable relationships, the ability to place a transaction in the appropriate Trust Context is even more critical. Trust context can be exchanged between participants, provided by a trusted 3rd party, or can exist in the form of a static entitlements policy against which transactions can be assessed.

SAML (Secure Assertion Markup Language) and XACML (XML Access Control Markup Language) are among the first such proposals for Trust Context standards.

SAML

SAML (Security Assertions Markup Language) is an initiative of OASIS (Organization for Advancement Structured Information Sciences) to provide a standard way to define user authentication, authorization, entitlements and profile information in XML documents. For both B2B and B2C, a single 'transaction' can often now be distributed across multiple companies, multiple Web sites, and multiple marketplaces, all of which may have their own authentication and authorization schemes. Companies need a standard open framework that will enable them to build trust chains across company boundaries, across heterogeneous platforms, and across multiple vendor solutions.

As its name suggests, SAML will allow business entities to make assertions regarding the identity, authorizations, and entitlements of a subject to other entities, which may be a partner company, another enterprise application etc. These assertions will be passed as XML documents, either pushed from the Asserting Party to the Relying Party or pulled from the Asserting Party to the Relying Party.

Consider the following scenario. A User A has authenticated to a B2B marketplace Exchange A. Part of the transaction that User A wishes to perform requires functionality not found amongst the members of Exchange A, but available from members of Exchange B (quite possible with vertical specific marketplaces). Before Exchange B can act on the behalf of User A and mediate the interaction with an appropriate supplier, it requires proof of the identity and credit rating of User A. Since Exchange A has already authenticated User A, and can attest to their credit rating, it provides to Exchange B a SAML assertion to that extent. Now trustful of the identity and credit worthiness of User A., Exchange B either provides an appropriate list of its members for User A to choose from or automatically selects one based on the SAML information.

The following XML listing is the entitlement message that would be passed from Exchange A to Exchange B. This XML assertion could be either bundled up with original XML transaction message or sent separately, either pushed or pulled. This example uses XML markup from the S2ML (Security Services Markup Language) proposal because the OASIS Technical Committee

has not yet defined the SAML XML Schema at the time of writing. S2ML is one of the references being considered by the OASIS Technical Committee.

```
<?xml version="1.0"?>
<Entitlement>
 <Id>urn:988876</Id>
 <Issuer>ExchangeA</Issuer>
 <Date>2000:12:23:12.00</Date>
 <Audiences>all</Audiences>
 <DependsOn>id4558999</DependsOn>
 <AzData>
 <cr:CreditRating>AAA+</cr:CreditRating>
 </AzData>
 <dsig:Signature>j&6fhl$3kppsdf</dsig:Signature>
</Entitlement>
```

This entitlement describes the credit rating for a previously authenticated party. The issuer signs the entitlement to bind its identity to the entitlement assertion.

Entrust is a member of the OASIS Security Services TC working on SAML standardization and co-chair of the Protocols and Core Assertions sub-groups. For a complete discussion of Entrust's authorization strategy, please refer to the 'XML Strategy for Authorization' white paper also available from the Web Services Trust portion of the Entrust Web site.

XACML

XML Access Control Markup Language is a proposal for providing XML documents with an access control model and access control specification language. Similar to existing policy languages, XACML is a language oriented around triplets of object, subject, and action. The granularity of object reference is as fine as a single element within an XML document and the action primitive consists of four kinds of actions: read, write, create, and delete.

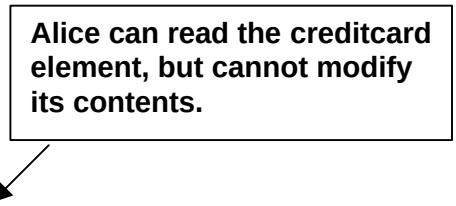
XACML is a language to describe element-wise access control policies. An object represents a single element or a set of elements in a target XML document. These elements are specified through an XPATH expression.

As an example, for an XML Purchase Order document structured like:

```
<?xml version="1.0"?>
<purchaseorder>
 .
  <creditcardnumber>123 456 789</creditcardnumber>
 .
</purchaseorder>
```

An XACML policy specifying that Alice has read but no write privileges for the credit card information might look like:

```
<?xml version="1.0"?>
<policy>
 <xacl>
 <object href=" creditcardnumber "/>
 <rule>
  <acl>
 <subject>
   <uid>Alice</uid>
 </subject>
 <action name="read" permission="grant"/>
  <action name="write" permission="deny"/>
 </acl>
 </rule>
 </xacl>
</policy>
```



Entrust has a reference implementation of an earlier proposal for an XML-based access control policy for XML documents available on its 'Web Services Trust' Web site. The demonstration illustrates how different healthcare roles, e.g. doctor, nurse, and patient, can be presented with different views, based on their access rights as defined in the XML policy, of an XML patient record.

There is currently a proposal for the formation of an OASIS Technical Committee to standardize XACML. Entrust expects to play an active role in the standardization of XACML through participation in this OASIS TC.

Conclusion

The Web Services model for business transactions enables applications to dynamically 'find and bind' to distributed components over the Internet in order to achieve some business transaction. The possibly high dollar value and sensitive nature of these transactions requires a comprehensive security infrastructure. Any such security infrastructure must address two issues fundamental to Web Services; the transactions will, for the most part, occur over the inherently insecure Internet and can occur between participants with no previous history on which trust could be based. The Web Services Trust Framework from Entrust addresses these issues, as well as the core requirements of authentication, authorization, confidentiality, integrity, and support for non-repudiation, through an extensible, standards-based solution set of products and technologies. The Web Services Trust Framework will allow Entrust's customers to easily integrate trust into their applications and Web services in a standards-based, open, and flexible manner.