SAML Conformance Clause for AS4/ebMS Version 1.0

Committee Specification Draft 01 / Public Review Draft 01

04 September 2013

Specification URIs

This version:

http://docs.oasis-open.org/ebxml-msg/ebms-v3.0-saml-conformance/v1.0/csprd01/ebms-v3.0-saml-conformance-v1.0-csprd01.doc (Authoritative)
http://docs.oasis-open.org/ebxml-msg/ebms-v3.0-saml-conformance/v1.0/csprd01/ebms-v3.0-saml-conformance-v1.0-csprd01.html
http://docs.oasis-open.org/ebxml-msg/ebms-v3.0-saml-conformance/v1.0/csprd01/ebms-v3.0-saml-conformance-v1.0-csprd01.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/ebxml-msg/ebms-v3.0-saml-conformance/v1.0/ebms-v3.0-saml-conformance-v1.0.doc (Authoritative)
http://docs.oasis-open.org/ebxml-msg/ebms-v3.0-saml-conformance/v1.0/ebms-v3.0-saml-conformance-v1.0.html
http://docs.oasis-open.org/ebxml-msg/ebms-v3.0-saml-conformance/v1.0/ebms-v3.0-saml-conformance-v1.0.pdf

Technical Committee:

OASIS ebXML Messaging Services TC

Chairs:

Sander Fieten (sander@fieten-it.com), Individual
Makesh Rao (marao@cisco.com), Cisco Systems, Inc.

Editor:

Ian Otto (Ian.Otto@innovation.gov.au), Australian Dept. of Industry, Innovation, Climate Change, Science, Research & Tertiary Education

Related work:

This specification is related to:

- *OASIS ebXML Messaging Services Version 3.0 Part 1: Core Features*. 01 October 2007. OASIS Standard. http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/ebms_core-3.0-spec.html.
- *AS4 Profile of ebMS 3.0 Version 1.0*. 23 January 2013. OASIS Standard. http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/profiles/AS4-profile/v1.0/AS4-profile-v1.0.html.
- *Web Services Security: SOAP Message Security Version 1.1.1*. 18 May 2012. OASIS Standard. http://docs.oasis-open.org/wss-m/wss/v1.1.1/wss-SOAPMessageSecurity-v1.1.1.html.
- *Web Services Security SAML Token Profile Version 1.1.1*. 18 May 2012. OASIS Standard. http://docs.oasis-open.org/wss-m/wss/v1.1.1/wss-SAMLTokenProfile-v1.1.1.html.

- *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*. 15 March 2005. OASIS Standard. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

Abstract:

This document provides a specification as to how an ebMS3/AS4 MSH can support SAML in addition to the username/password and X.509 token profiles of WS-Security for authentication.

The usage of the username/password and X.509 token profiles of WS-Security are defined in some detail in ebMS3 and AS4. SAML is included in ebMS3 but is not defined in detail. AS4 does not discuss SAML.

This document will define the use of SAML in ebMS3 and apply that to the scenarios addressed by AS4.

Status:

This document was last revised or approved by the OASIS ebXML Messaging Services TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/ebxml-msg/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/ebxml-msg/ipr.php).

Citation format:

When referencing this specification the following citation format should be used:

[ebMS-saml-conformance]

SAML Conformance Clause for AS4/ebMS Version 1.0. 04 September 2013. OASIS Committee Specification Draft 01 / Public Review Draft 01. http://docs.oasis-open.org/ebxml-msg/ebms-v3.0-saml-conformance/v1.0/csprd01/ebms-v3.0-saml-conformance-v1.0-csprd01.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/policies-guidelines/trademark for above guidance.

Table of Contents

1 Introduction

This specification describes the applicability and usage of SAML Assertions (usually, but not exclusively obtained through WS-Trust) along-side Username Token and X.509 Token in securing ebXML Messaging Service (ebMS) messaging. SAML Assertions provide a more dynamic way of establishing and managing identity and roles than their counter-parts.

1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **[RFC2119]**.

1.2 Normative References

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

[EBMS3CORE] *OASIS Standard, OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features*, October 2007, http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/os/ebms_core-3.0-spec-os.odt

[EBMS3ADV] *OASIS ebXML Messaging Services Version 3.0: Part 2, Advanced Features. Committee Specification* 01, 19 May 2011., http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/part2/201004/ebms-v3-part2.odt

[EBMS3-AS4] *OASIS Standard, AS4 Profile of ebMS 3.0 Version 1.0*, 23 January 2013, http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/profiles/AS4-profile/v1.0/os/AS4-profile-v1.0-os.odt

[SAMLCoreV1] *Oasis Standard, E. Maler, P.Mishra, and R. Philpott (Editors), Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V1.1*, September 2003. https://www.oasis-open.org/committees/download.php/3406/

[SAMLCoreV2] *Oasis Standard, S. Cantor, J. Kemp, R. Philpott, E. Maler (Editors), Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005. http://docs.oasis-open.org/security/saml/v2.0/

[WSS-SAML] *OASIS Standard, Kelvin Lawrence, Chris Kaler (Editors), Web Services Security:SAML Token Profile 1.1*, 1 February 2006 http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf

[WSS111] Anthony Nadalin, et al, eds., *Web Services Security: SOAP Message Security 1.1.1*, May 2012. http://docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-SOAPMessageSecurity-v1.1.1-os.doc

[WS-Trust] *OASIS Standard, Anthony Nadalin, et al, eds.,WS-Trust 1.3*, March 2007. http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html

1.3 Non-Normative References

[WS-Trust14] *WS-Trust 1.4*. 25 April 2012. OASIS Standard incorporating Approved Errata. http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/errata01/os/ws-trust-1.4-errata01-os-complete.html

[WSPOLICY] A. Vedamuthu, et al, eds, *Web Services Policy 1.5*: Framework, 2007. http://www.w3.org/TR/ws-policy/

[WSSECPOL] A. Nadalin, et al, eds, *WS-SecurityPolicy 1.2*, 2007. http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.pdf

[XMLDSIG] Donald Eastlake, et al, eds, *XML-Signature Syntax and Processing*, 2002. http://www.w3.org/TR/xmldsig-core

[XMLENC] D. Eastlake, et al, *XML Encryption Syntax and Processing*, 2002. http://www.w3.org/TR/xmlenc-core

2 Background and Objectives

[EBMS3CORE] states that WS-Security is the mechanism that is used to secure messages, and then goes on to provide examples for X.509 Tokens and Username Tokens in specific can be used. The use of SAML is alluded to but not specified.

The purpose of this specification is to provide insight and guidance on how SAML should be used in the ebMS context.

The core SAML specifications cover two facets of SAML: SAML Assertions which are a representation of an identity, and the SAML Protocol which is a way that SAML Assertions can be exchanged to achieve various goals. SAML Assertions have a wide applicability outside of the SAML Protocol and it is this usage which is important in the specification. The SAML Protocol will not be discussed further in this document.

The versions of SAML in common use are SAML 1.1 and SAML 2.0. While syntactically different, the two specifications are semantically similar enough that they can be treated as being the same for the purposes of this specification.

2.1 Federated Identity Providers and Their Role in an eBusiness Messaging Framework

By and large, ebMS is used to exchange messages in communities where there is a relatively stable membership. The cost of changing passwords occasionally or rolling of X.509 credentials as they expire can be effectively managed albeit with some cost.

ebMS however is starting to be used in communities where there is a large number of clients talking to one or more hubs. In such communities, use of an Identity Provider can reduce the burden on both the client and the hub.

There are a numerous benefits including:

- when the hub needs to roll its X.509 credentials, it only has to notify the Identity Provider, not each of the clients.
- identity providers isolate the hub from technology changes. Regardless of how the client authenticates to the identity provider, the hub will receive identity as a standard SAML Assertion.
- the identity information may be supplemented by additional information, for authorization purposes.

3 SAML Tokens in the Context of ebMS 3.0

A SAML Token (SAML Assertion) can be used to convey identity in a number of different circumstances including Web Services Security **[WSS111]**. **[WSS-SAML]** defines the ways in which a SAML Token can be employed to secure Web Services.

Implementations MUST support SAML 2.0 as defined in **[SAMLCoreV2]**. As SAML 1.1 is still in wide use, implementations MAY choose to support SAML 1.1 Assertions as defined in **[SAMLCoreV2]**.

3.1 SAML Subject and Attributes

A SAML Subject identifies an end entity in an assertion, normally the requester of the Assertion. The SAML Subject is typically a unique and persistent identifier. The identifier is also normally opaque so it only has meaning in the context of a broader set of systems sharing identity and authorization information.

SAML Attributes (in WS-Trust parlance these are referred to as Claims) are attributes of the identity. An Assertion contains zero or more Attributes that describe the subject.

One or more of the attributes may convey identity in a form that can be verified against the Sender/Receiver or Producer/Consumer of the message.

The values of these SAML Attributes MAY be used as referred to in **[EBMS3CORE]** (7.12.7. Persistent Authorization) to authorize access.

3.2 Types of SAML Tokens

SAML Assertions can be issued as Holder-Of-Key, Sender-Vouches, or Bearer tokens. Bearer tokens are not supported by the **[WSS-SAML]**. Sender-Vouches tokens may be useful in some multi-hop scenarios, but are outside the scope of this version of the specification. (In Sender-Vouches, the receiver must trust the sender that they are authorised to use the attached SAML token. It is not cryptographically bound to them in any way.)

A Holder-Of-Key token contains a proof key that can be used to verify the originator of a message is the legitimate owner of the token.

If the proof key is a symmetric key, then it MUST be wrapped (using the public key of the recipient) for the intended recipient of the token. Only the recipient can validate the message.

If the proof key is the public half of an asymmetric key pair, then it can be included in the assertion unencrypted. In this case, anyone can validate a signature made by the message initiator who controls the private half of the key pair.

a) An Initiating MSH MUST be able to employ a Holder-Of-Key token with a symmetric proof key.

b) A Responding MSH MUST be able to verify a Holder-Of-Key token with a symmetric or asymmetric proof key.

3.3 Proof Key Applicability to Multi-Hop Scenarios

In multi-hop scenarios where the intermediaries are purely routing on header information (specified in **[EBMS3ADV]**), if a symmetric proof key is used, it MUST be wrapped for the final recipient.

Alternately, in circumstances where end to end signatures are required on information transiting a multi-hop chain and intermediaries need to validate the security on the message, an asymmetric proof key MUST be employed.

In this case, the SAML token acts as an X.509 certificate equivalent, carrying the public key tied to the identity. There are the added benefits that identity information can be enriched with attributes and that revocation checks are not required due to the SAML Token's limited lifetime.

In single hop scenarios or scenarios where intermediaries only perform a routing function, either symmetric or asymmetric proof keys may be employed.

3.4 Audience and SubjectConfirmation

It is important to note the following regarding the SAML token as obtained from an Identity Provider:

- When symmetric keys are employed, SAML tokens need to be targeted at a particular SOAP receiver endpoint. When communicating with the IdP to obtain the token, the SOAP sender MUST supply an appropriate identifier for the SOAP receiver.
- The SubjectConfirmation element of the SAML token MUST contain a copy of the proof key encrypted for the SOAP receiver.
- By convention, this is normally the SOAP endpoint of the receiver, but it MAY be any URI that logically denotes the SOAP receiver.
- This URI is placed into the Audience element of the SAML token and it is normally used by the IdP to determine how to suitably wrap the proof key for consumption by that SOAP Receiver.

For symmetric proof keys, this is essential to preserve the security of the key.

3.5 Security Elements in the SOAP Header

[EBMS3CORE] defines the option of having two Security elements in the SOAP header, one that secures the message, and one that provides an authorization token.

Unlike username/password, a Holder-Of-Key SAML token must exercise its proof key to prove the Initiating MSH is the legitimate bearer of the token. This is beyond the scope of the authorization Security element.

The SAML token MUST be used in the main Security element when employed.

Note: The authorization Security element MAY still be employed if the SAML token is used for authentication only.

3.6 SAML Attributes for MPC Authorization

In ebMS3, messages may be placed into Message Partition Channels (MPCs) for collection. Each MPC can have individual authorization for collecting entities.

A SAML token typically contains a number of attributes, name-value pairs, representing facts about the subject of the token.

An MSH MAY authorize access to an MPC on the basis of a set of attributes from the SAML token.

For example[1]:

- SAML Tokens from a particular Identity Provider may contains attributes: BusinessId and Region
- A particular MPC may require a BusinessId of Supplier496 and Region of NorthAmerica for access.
- That MPC would only allow access to entries where both attributes are present in the token with the required values.

[1] Note: This example is non-normative.

3.7 SAML Token Lifetime

While SAML tokens are similar to X.509 certificates in many respects, a major difference is that SAML tokens are far more short lived. Depending on the policy of the Identity Provider and the lifetime requested by subject, lifetimes will normally be in the range of 5 minutes to 8 hours at most. The short lifetime has benefits in that the revocation checks required by X.509 are not needed, but does add other complications.

Where SOAP messages are held by an MSH awaiting transmission for a period longer than the SAML token lifetime, the implementation MUST refresh the token and update the signature prior to transmission.

3.8 Sample Message

The following example is extracted from the ebMS3 core specification and the X509 Digital Signature is replaced with a SAML based digital signature. Changed regions have been highlighted.

Note: SAML is normally only used for authentication in SOAP Requests. Where Authentication/Non-Repudiation of a SOAP Response is required, it is normal to use an X.509 Signature.

```
Mime-Version: 1.0
Content-Type: text/xml
Content-Transfer-Encoding: binary
SOAPAction: ""
Content-Length: 7205

<?xml version="1.0" encoding="UTF-8"?>
<S12:Envelope xmlns:S12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/ns/core/200704/
    http://docs.oasis-open.org/ebxml-msg/ebms/v3.0/core/ebms-header-3_0-200704.xsd" >
    <S12:Header xmlns:eb="http://docs.oasis-open.org/ebxml-
msg/ebms/v3.0/ns/core/200704/">

        <eb:Messaging id="ebMessage" S12:mustUnderstand="true">
            <eb:UserMessage>
                <eb:MessageInfo>
                    <eb:Timestamp>2006-10-31T17:36:20.656Z</eb:Timestamp>
                    <eb:MessageId>UUID-2@msh-server.example.com</eb:MessageId>
                    <eb:RefToMessageId>UUID-1@msh-server.example.com</eb:RefToMessageId>
                </eb:MessageInfo>
                <eb:PartyInfo>
                    <eb:From>
                        <eb:PartyId>uri:msh-server.example.com</eb:PartyId>
                        <eb:Role>http://example.org/roles/Buyer</eb:Role>
                    </eb:From>
                    <eb:To>
                        <eb:PartyId type="someType">QRS543</eb:PartyId>
                        <eb:Role>http://example.org/roles/Seller</eb:Role>
                    </eb:To>
                </eb:PartyInfo>
                <eb:CollaborationInfo>
                    <eb:AgreementRef>http://msh-
server.example.com/cpa/123456</eb:AgreementRef>
                    <eb:Service type="someType">QuoteToCollect</eb:Service>
                    <eb:Action>NewPurchaseOrder</eb:Action>
                    <eb:ConversationId>2a81ffbd-0d3d-4cbd-8601-
d916e0ed2fe2</eb:ConversationId>
                </eb:CollaborationInfo>
                <eb:MessageProperties>
                    <eb:Property name="ProcessInst">PurchaseOrder:123456</eb:Property>
                    <eb:Property name="ContextID">987654321</eb:Property>
                </eb:MessageProperties>
                <eb:PayloadInfo>
                    <eb:PartInfo href="#enc">
                        <eb:Description xml:lang="en-US">PO Image</eb:Description>
                    </eb:PartInfo>
                </eb:PayloadInfo>
            </eb:UserMessage>
        </eb:Messaging>
```



```
        <wsse:Security S12:mustUnderstand="true"
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
      <saml:Assertion MajorVersion="1" MinorVersion="1" AssertionID="_398c2fae-4178-
4ff4-ac59-95d491dbbdf6"
               Issuer="Example Security Token Service" IssueInstant="2013-05-
16T02:09:29.645Z"
                xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
        <saml:Conditions NotBefore="2013-05-16T02:09:29.647Z" NotOnOrAfter="2013-05-
16T02:39:29.647Z">
          <saml:AudienceRestrictionCondition>
            <saml:Audience>https://soapreceiver.example.org.au</saml:Audience>
          </saml:AudienceRestrictionCondition>
        </saml:Conditions>

        <saml:AttributeStatement>
          <saml:Subject>
            <saml:NameIdentifier>
              urn:example.org:id:1204567890
            </saml:NameIdentifier>
            <saml:SubjectConfirmation>
              <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:holder-of-
key</saml:ConfirmationMethod>
               <!-- This is the proof key wrapped for the Responding MSH -->
              <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                <e:EncryptedKey xmlns:e="http://www.w3.org/2001/04/xmlenc#">
                  <e:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
oaep-mgf1p">
                    <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></DigestMethod>
                  </e:EncryptionMethod>
                  <!-- This is the certificate of the Responding MSH -->
                  <KeyInfo>
                    <o:SecurityTokenReference
                    xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
                      <X509Data>
                        <X509IssuerSerial>
                          <X509IssuerName>CN=Example CA,  O=Example CA Organisation,
C=AU</X509IssuerName>

<X509SerialNumber>37310890721155718122974868787627716901</X509SerialNumber>
                        </X509IssuerSerial>
                      </X509Data>
                    </o:SecurityTokenReference>
                  </KeyInfo>
                  <e:CipherData>

<e:CipherValue>uW2St4BD9+lZzSGbSkvhqIkCoMwVlf3qDJl2X4Nj8bP8LQxJpchQugHKV+7y+8k1vrVxPPapx
ka7aWscDvGbmHT9cxaAqnNtTuK2R7yo1i22yNxSa3us5l1VHLFB447tAf/tQ/OQPsD4myTqad2+LLoDT6lS0CrJO
/Ue+WMLNzI=</e:CipherValue>
                  </e:CipherData>
                </e:EncryptedKey>
              </KeyInfo>
            </saml:SubjectConfirmation>
          </saml:Subject>
          <saml:Attribute
           AttributeName="A_Relevant_Attribute_or_Claim"
AttributeNamespace="http://example.org/2008/06/identity/claims">
            <saml:AttributeValue>  Relevant Attribute Value
            </saml:AttributeValue>
          </saml:Attribute>
        </saml:AttributeStatement>

        <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
          <SignedInfo>
            <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"></CanonicalizationMethod>
            <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"></SignatureMethod>
            <Reference URI="#_398c2fae-4178-4ff4-ac59-95d491dbbdf6">
```



```
              <Transforms>
                <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"></Transform>
                <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"></Transform>
              </Transforms>
              <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></DigestMethod>
              <DigestValue>0GEhLfUS2pGSp4ZziJJsV9VbeW8=</DigestValue>
            </Reference>
          </SignedInfo>

<SignatureValue>DpZX6V4Wn2RI0+a3jug3H5gfa4MZiOGSQ/rfsLHkE0X/HgzV4cZDl4wFtPqBCdm9eyByNtDj
zaSKRKT3Md5LgANxMY5deJGJvPmGyQSfrSMrCUCPv5iktaCQEJSpFS+R5KLdSdBkJuFaT6JAYE2CfF6BVk0LGP8L
hW/Z6qFfzrA=</SignatureValue>
          <KeyInfo>
            <X509Data>
              <X509Certificate> <!-- Certificate of the STS -- trusted by the responding
MSH --> </X509Certificate>
            </X509Data>
          </KeyInfo>
        </Signature>
      </saml:Assertion>

      <wsse:BinarySecurityToken
                EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary"
                ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
token-profile-1.0#X509v3"
                wsu:Id="encryptionCert"><!-- This may be the same certificate as in the
green section above. It MUST be a certificate to which the Responding MSH has the
private key --> </wsse:BinarySecurityToken>
      <enc:EncryptedKey xmlns:enc="http://www.w3.org/2001/04/xmlenc#" >
                <enc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
1_5"
                    xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"/>
                <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                    <wsse:SecurityTokenReference>
                        <wsse:Reference URI="#encryptionCert"/>
                    </wsse:SecurityTokenReference>
                </KeyInfo>
                <CipherData xmlns="http://www.w3.org/2001/04/xmlenc#">

<CipherValue>F3HmZ2Ldyn0umLCx/8Q9B9e8OoslJx9i9hOWQjh6JJwYqDLbdg0QVFiVT1LVjazlThS9m9rkRtp
khCUIY1xjFKtDsuIIAW8cLZv7IHkVoDtQ7ihJc8hYIlEESX9qZN65JgyAa3BYgW9ipjGHtNgZ9RzUdzKdeY74DFm
27R6m8b0=</CipherValue>
                </CipherData>
                <ReferenceList xmlns="http://www.w3.org/2001/04/xmlenc#">
                    <DataReference URI="#enc"/>
                </ReferenceList>
        </enc:EncryptedKey>
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
                    <!-- As this example uses a symmetric key, we must use an
appropriate signature algorithm -->
                    <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"/>
                    <ds:Reference URI="#ebMessage">
                        <ds:Transforms>
                            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
                        </ds:Transforms>
                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>Ae0PLUKJUnUyAMXkLQD/WwKiFiI=</ds:DigestValue>
                    </ds:Reference>
                    <ds:Reference URI="#body">
                        <ds:Transforms>
                            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
```



```
                        </ds:Transforms>
                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <ds:DigestValue>kNY6X7LnRTwxXXBzSw07tcA0KSU=</ds:DigestValue>
                    </ds:Reference>
                </ds:SignedInfo>
                <ds:SignatureValue>

T24okA0MUh5iBNMG6tk8QAKZ+lFMmY1rcPnkOr9j3fHRGM2qqUnoBydOTnClcEMzPZbnlhdN

YZYmab1lqa4N5ynLjwlM4kp0uMip9hapijwL67aBnUeHiFmUau0x9DBOdKZTVa1QQ92106ge
                    j2YPDt3VKIlLLT2c8O4TfayGvuY= </ds:SignatureValue>
                <ds:KeyInfo>
                    <!-- As it would be for an X.509 certificate, There is a security
token reference in the signature referencing the SAML token as the source of the signing
key. -->
                    <wsse:SecurityTokenReference
                        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd"
                        k:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-
token-profile-1.1#SAMLV1.1"
                    xmlns:k="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-
1.1.xsd">
                          <wsse:KeyIdentifier
                       ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.0#SAMLAssertionID">
                              _398c2fae-4178-4ff4-ac59-95d491dbbdf6
                     </wsse:KeyIdentifier>
                      </wsse:SecurityTokenReference>
                </ds:KeyInfo>
            </ds:Signature>

        </wsse:Security>

    </S12:Header>

    <S12:Body wsu:Id="body"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">
        <EncryptedData Id="enc" Type="http://www.w3.org/2001/04/xmlenc#Content"
            xmlns="http://www.w3.org/2001/04/xmlenc#">
            <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
cbc"/>
            <CipherData>

<CipherValue>tjOgUPMmQwd6hXiHuvl42swqv4dTYiBfmg8u1SuFVRC3yfNlokshvoxs1/qQoqN1prDiSOxsxsF
vg1la7dehjMWb0owuvU2de1eKr5KPcSApnG+kTvNrtg==</CipherValue>
            </CipherData>
        </EncryptedData>
    </S12:Body>

</S12:Envelope>
```

Note in the following example:

- A symmetric proof of possession/subject confirmation key has been employed in this example although asymmetric keys could be used if required. Both are valid for Holder of Key tokens.
- When a symmetric key is used, it must be encrypted for the receiver by the token issuer as is normal practice for Holder of Key Tokens. It is normally encrypted by the public key from an X.509 certificate provided to the token issuer by the SOAP Receiver.
- Only a single attribute/claim and value have been included in the SAML token, but zero or more may be provided in the general case. When no claims are provided, the receiver SHOULD make authorization decisions based on previous knowledge of the subject id.
- With respect to the example in **[EBMS3CORE]** in section 7.9.1, the SAML Token has replaced the BinarySecurityToken for the X.509 Certificate for the signer.
  from:
  `<wsse:BinarySecurityToken`


```
      EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soap-message-security-1.0#Base64Binary"
      ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-x509-token-profile-1.0#X509v3"
      wsu:Id="signingCert">...</wsse:BinarySecurityToken>
```

to:

```
 <saml:Assertion …> … </saml:Assertion>
```

- The signing algorithm has been changed to reflect the symmetric signing key used.
  from:

```
  <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"/>
```

  to:

```
 <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-
sha1"/>
```

- The `SecurityTokenReference` in the message signature has been changed to refer to the SAML token.
- Encryption is unchanged from the example in **[EBMS3CORE]**. The Responding MSH's X.509 credential SHOULD be used where secure transport is unsuitable or unavailable.

4 Acquiring SAML Tokens

A SAML Token MUST be acquired from an Identity Provider/Security Token Service that is acceptable to both the Initiating MSH and the Responding MSH.[2]

Implementations of this specification SHOULD support **[WS-Trust]** for token acquisition. Additional mechanisms MAY be supported. It is recommended that **[WS-Trust14]** be supported as and when implementations become more commonplace.

4.1 Acquiring SAML Tokens with WS-Trust

The Initiating MSH acquiring a token by making a **[WS-Trust]** call to the Identity Provider MUST provide:

a) The token type to be acquired (i.e. SAML 2.0);

b) The Responding MSH that the token applies to (by convention, this is the endpoint of the MSH, but it can be any URI that logically identifies the Responding MSH that the Identity Provider knows about.); and

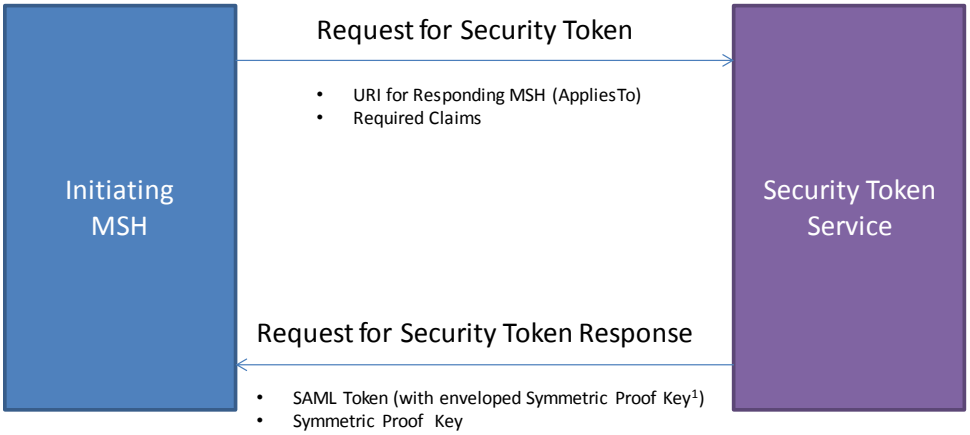
c) A list of the claims that the Initiating MSH wishes to supply to the Responding MSH. (Advertised to the Initiating MSH via the Responding MSH PModes.)

Additionally, the Key Type of the proof key MAY be specified although it will default to symmetric if unspecified.

Note that in accordance with **[WS-Trust]** if asymmetric is specified, the public key must be provided by the Initiating MSH and the corresponding private key must be exercised to provide ownership in the token request.

In the default case or when a symmetric key is explicitly requested, the Identity Provider will normally generate a key and return it directly to the Initiating MSH along with a copy of the key wrapped for the Responding MSH inside the SAML token.

[2] Note that SAML Authentication is normally one way. The return path from the Responding MSH to the Initiating MSH is either secured using X.509 or relies on transport security.



Additional options are available in **[WS-Trust]** for symmetric key generation and MAY used if desired.

Support for acquiring tokens using **[WS-Trust14]** SHOULD be implemented to maximize interoperability with newer product sets as they become available.

4.2 Responding MSH Registration with a WS-Trust Identity Provider

The Responding MSH must be registered with the **[WS-Trust]** Identity Provider. The registration process is not defined by the specification beyond stating that:

a) The Responding MSH MUST supply the Identity Provider with a URI that logically or physically denotes the Responding MSH as noted above in section 4.1b.

b) The Responding MSH MUST supply the Identity Provider with an X.509 Certificate with which to wrap the proof key.

Additionally, the Responding MSH MAY supply to the Identity Provider (where supported by the Identity Provider):

c) A list of claims required and/or desired by the Responding MSH

d) The type of token required by the Responding MSH (e.g. SAML 2.0)

5 Error Code Mapping

Introducing SAML as an authentication mechanism adds an additional party into the authentication process and while the nature of the errors does not really change, the locations where they occur does.

Most identity providers will use WS-Trust to issue SAML identity tokens in the web service context although the following mapping applies regardless of the protocol used.

The following table contains the relevant errors to SAML authentication:

EBMS:0005	ConnectionFailure
EBMS:0101	FailedAuthentication
EBMS:0103	PolicyNoncompliance

Error situations in the token issue process and use of SAML tokens will map to the following ebMS errors. The MSH MUST generate the ebMS error specified in the table below. Other errors MUST be processed as defined in the P-Mode (**PMode[1].ErrorHandling).**

Situation	**EBMS**			**Type**
	0005	**0101**	**0103**	
Failure of SOAP Sender to connect to STS	X			Issue
STS could not authenticate sender		X		Issue
STS could not provide Mandatory claims for Sender			X	Issue
Token signature does not verify		X		Use
Token issuer not recognized by receiver		X		Use
Token Expired		X		Use
Mandatory claims missing from Token			X	Use
Token subject is unknown to Receiver[3]			X	Use

[3] Just because an Identity Provider can authenticate an entity, that does not mean that the entity is registered with the SOAP receiver for the purposes of authorisation and access.

6 Processing Modes for SAML Security

The following PModes are specific to a SAML implementation. These PModes are specific to the MSH Responder and indicate what the MSH Initiator requires in order to communicate with that responder.

These PModes in whole or in part may be expressed through various mechanisms specific to the implementation.

PMode[1].Security.SAML.Version: The value of this parameter is a list of the versions of SAML Tokens supported. At present, SAML20 MUST be supported. SAML11 MAY be supported.

PMode[1].Security.SAML.RegisteredIdPs: The value of this parameter is a list of the IdPs acceptable to the SOAP Receiver. Each IdP entry consists of the URL of the IdP and the URI by which the IdP knows the SOAP Receiver. Typically this URI is the URL of the SOAP Receiver endpoint, but it may be any valid URI.

PMode[1].Security.SAML.MandatoryAttributes: The value of this parameter is a list of SAML Attributes describing the subject that are required in the SAML Assertion.

PMode[1].Security.SAML.OptionalAttributes: The value of this parameter is a list of SAML Attributes describing the subject that should be provided in the SAML Assertion, but are not required.

PMode[1].Security.SAML.KeyType: The value of this parameter denotes the type of proof key required in the SAML Assertion. The key type may be Symmetric or Asymmetric.

The following PModes parameters are used in message pulling to authorize access to that MPC:

PMode[1].Initiator.Authorization.SAML.AttributesAndValues: This parameter contains a list of SAML attributes that messages for a particular MPC MAY be authorized on. All in the list MUST be provided.

The following PModes are unchanged from **[EBMS3CORE]** and are apply to SAML in the same way they apply to X.509:

PMode[1].Security.X509.Signature.HashFunction: The value of this parameter identifies the algorithm that is used to compute the digest of the message being signed. The definitions for these values are in the **[XMLDSIG]** specification.

PMode[1].Security.X509.Signature.Algorithm: The value of this parameter identifies the algorithm that is used to compute the value of the digital signature. The definitions for these values are found in the **[XMLDSIG]** or **[XMLENC]** specifications.

Encryption of elements should use the PModes defined in **[EBMS3CORE]** section D.3.6, **PMode[1].Security. X509.Encryption.***.

7 Conformance

This conformance section should be read in conjunction with [EBMS3-AS4], as it is an adjunct to that specification.

7.1 AS4 ebHandler Conformance Profile Supplement

This section defines the additions to Section 2.1 of **[EBMS3-AS4]**.

7.1.1 Security

The following additional Security features MUST be supported:

1. Support for acquiring SAML tokens via **[WS-Trust]** [4.1].
2. Support for securing outgoing SOAP requests with SAML 2.0 Security Tokens in accordance with **[WSS-SAML]** in the primary Security Element [3.5]
3. Support for receiving and validating SOAP requests secured by SAML 2.0 Security Tokens in accordance with **[WSS-SAML]** in the primary Security Element [3.5]
4. Support for sending and receiving Holder-Of-Key tokens with symmetric keys [3.2-a].
5. Support for receiving Holder-Of-Key tokens with asymmetric keys as per [3.2-b]

7.2 AS4 Light Client Conformance Profile Supplement

The following additional Security features MUST be supported:

1. Support for acquiring SAML tokens via **[WS-Trust]** [4.1].
2. Support for securing outgoing SOAP requests with SAML 2.0 Security Tokens in accordance with **[WSS-SAML]** in the primary Security Element [3.5]
3. Support for sending Holder-Of-Key tokens with symmetric keys [3.2-a].

7.3 AS4 Minimal Client Conformance Profile Supplement

The following additional Security features MUST be supported:

1. Support for acquiring SAML tokens via **[WS-Trust]** [4.1].
2. Support for securing outgoing SOAP requests with SAML 2.0 Security Tokens in accordance with **[WSS-SAML]** in the primary Security Element [3.5]
3. Support for sending Holder-Of-Key tokens with symmetric keys [3.2-a].

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Sander Fieten (sander@fieten-it.com), Individual
Makesh Rao (marao@cisco.com), Cisco Systems, Inc.
Ian Otto (Ian.Otto@innovation.gov.au), Australian Dept. of Industry, Innovation, Climate Change, Science, Research & Tertiary Education
Theo Kramer (theo@flame.co.za), Flame Computing Enterprises
Dale Moberg (dmoberg@us.axway.com), Axway Software
Pim van der Eijk (pvde@sonnenglanz.net), Sonnenglanz Consulting
Jacques Durand (jdurand@us.fujitsu.com), Fujitsu America Inc.
Malcolm Young, Australian Dept. of Industry, Innovation, Climate Change, Science, Research & Tertiary Education
Michael Leditscke, Australian Taxation Office

Appendix B. How SAML is Beneficial (Non-Normative)

SAML is sometimes presented as being more complex than X.509 and Username/Password. Mostly, it is that it is newer and hence less well supported and understood. It is more powerful and has long term benefits that will become apparent in this appendix.

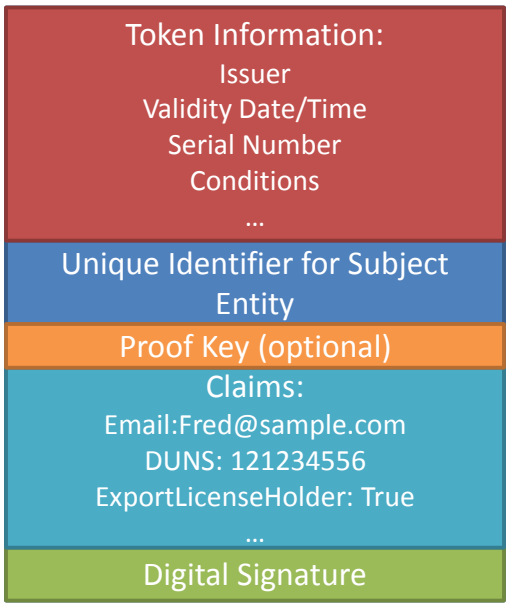
SAML can be used in small fixed communities, but where SAML comes in to its own is in *many to few* scenarios. SAML is particularly suited to client polling hub scenarios.

While SAML is normally used for securing SOAP requests, X.509 is normally used for securing the SOAP responses. Hubs are typically well known, carry out a fixed role, and are not dynamically configured so X.509 is appropriate. Clients of hubs are more dynamic and change roles over time so SAML is more appropriate.

B.1 What is a SAML Token?

A SAML Token is a relatively small XML document fragment which certifies the identity and attributes of an entity transaction on the internet.

In common scenarios, this entity may be a person, a business representative, or a business back end system.



SAML Tokens are normally ephemeral. They convey identity for a short period of minutes to hours. They can be used in a number of contexts, from web based single sign on to web services.

A SAML token normally contains a subject which will be a unique identifier for the entity. It is typically used for tracking and authorization purposes where the SOAP receiver has its own authorization database.

A SAML token also contains attributes of the entity. In WS-Trust parlance, these attributes are referred to as claims. A claim has a name and a value. Claims typically contain verified information about the entity such as email address. They could contain information such as the DUNS number of a business entity or other common registration number for a particular community.

SAML attributes can also contain role information for an entity. A business representative may carry a claim of “Authorized Purchasing Officer” for example.

The SAML token has an XML-DSIG signature of a trusted identity provider covering it to certify the validity and correctness of the information contained in the token.

When SAML tokens are used in web services, they have a cryptographic key, the Proof Key, which binds the identity of the owner to messages they are sending in the same way that an X.509 Certificate does.

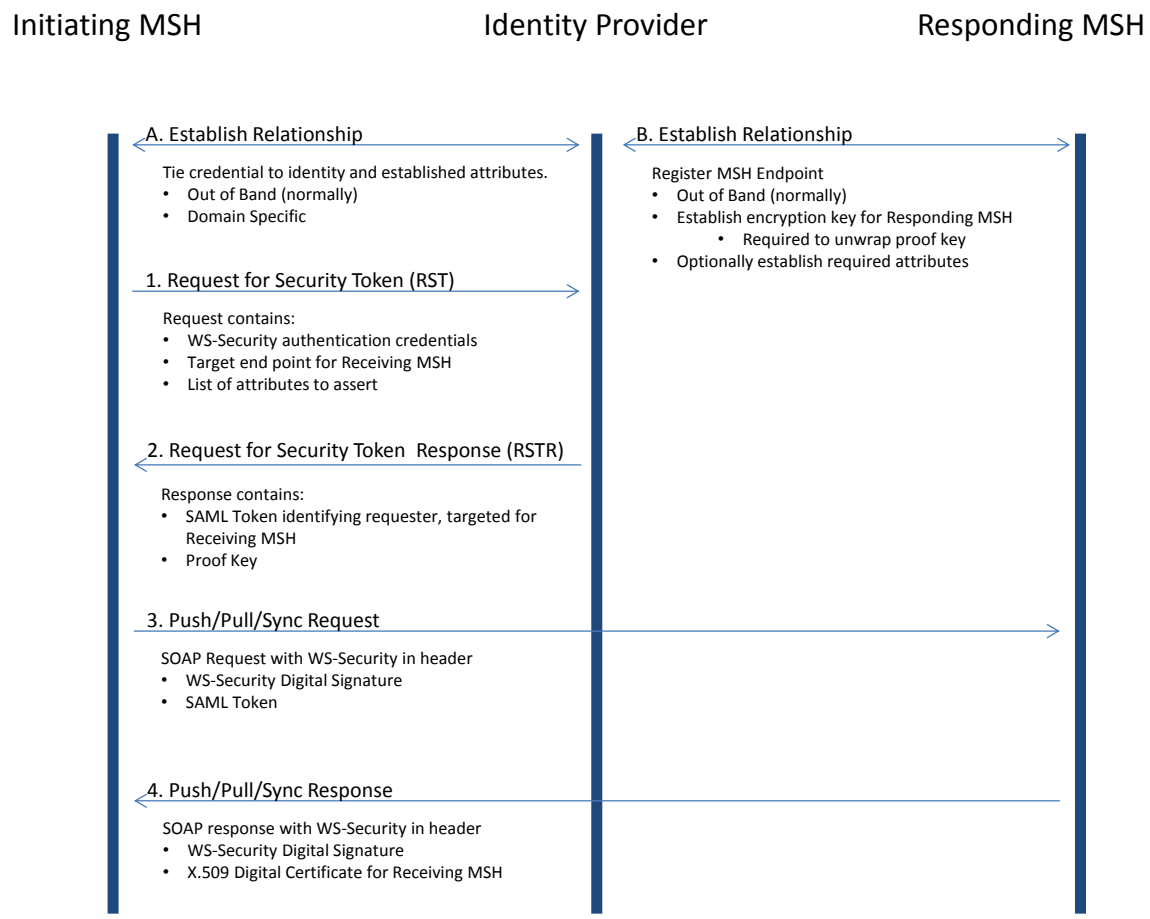
Unlike an X.509 Certificate, because SAML tokens are normally relatively short lived and are populated with up to date information at the time they are issued (typically within a few minutes of performing a transaction), there is no need for a revocation mechanism.

Based on the sensitivity of a transaction, a token receiver can always reject a token based on authentication event that the token was based on being outside its tolerance.

B.2 Obtaining a SAML Token

In web services scenarios, SAML Tokens are provided as they are required from an Identity Provider, typically using the WS-Trust protocol. In WS-Trust parlance, the Identity Provider is referred to as a Security Token Service (STS)

The following diagram shows a typical flow for a WS-Trust based identity provider:



Prior to any transactions taking place a small amount of registration needs to take place. For the simple case where only one identity provider involved, it will involve something like the following:

A. The Initiating MSH is configured with a credential that has been obtained from an Identity Provider. Obtaining such a credential typically requires some sort of Evidence of Identity for the registrant and a Proof of Association with the business.
   Typically validated attributes associated with the credential will be held by the credential provider. (For example,a unique business identifier.)
B. The Responding MSH registers with the Identity Provider as a relying party. At a minimum, the relying party registration requires:
   - An endpoint for the service they are hosting

- An X.509 Certificate that can be used for wrapping proof keys so that only the relying party can extract them.

Optionally a set of required attributes (claims) may be registered.

In return, the Identity Provider provides an X.509 Certificate that will be used to verify the siugnatures on SAML Assertions (SAML Tokens) supplied by the Identity Provider.

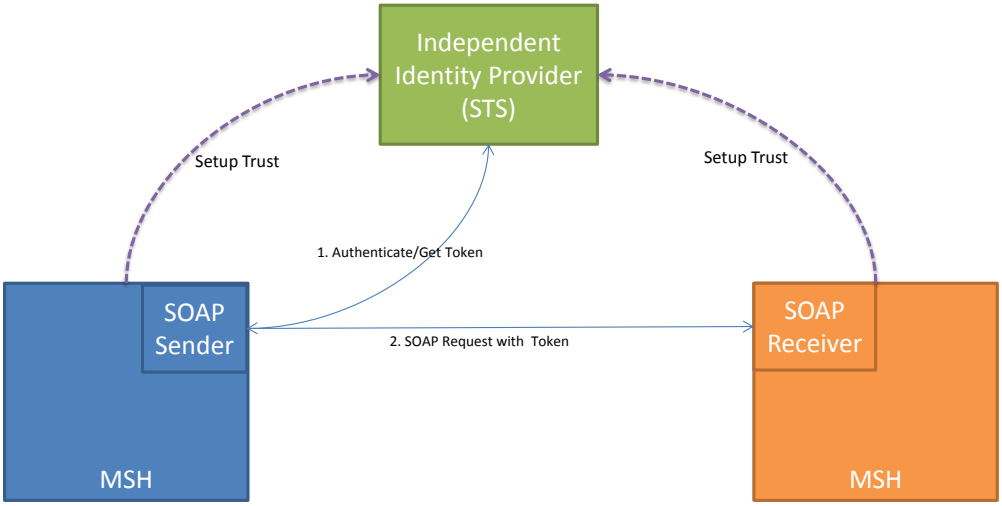
At transmission time, the following steps would normally be taken:

1. The Initiating MSH makes a WS-Trust call to the Identity Provider in the form of a Request Ssecurity Token message. The call includes:
   - The endpoint that the Initiating MSH is targeting. This is supplied in the "AppliesTo" element of the request. (the Responding MSH)
   - Authentication information that verifies the identity of the Initiating MSH to the Identity Provider. (Typically through a WS-Security compliant mechanism.)
   - A list of the attributes (claims) that the Initiating MSH wishes to have asserted to the Responding MSH.
2. The Identity Provider responds with a Request for Security Token Response message which includes:
   - A SAML Assertion (SAML Token) including:
     - the identity of the Initiating MSH,
     - the required attributes (claims),
     - the key for verifying signatures made by the Initiating MSH, encrypted for the Responding MSH
   - A Proof Key to be used by the Initiating MSH to sign requests to the Responding MSH.
3. The Initiating MSH makes a SOAP call to the Responding MSH including:
   - The standard ebMS3 SOAP headers appropriate for the request
   - The WS-Security header including:
     - The message signature, signed with the Proof Key from step 2;
     - A reference to the SAML Token in the signature's KeyInfo;
     - The SAML Token
   - The standard ebMS3 SOAP Body
4. The Responding MSH returns a SOAP response to the Sending MSH as it would normally in an X.509 scenario, signing the response with its X.509 Certificate.

Note that SAML tokens are typically re-used for a period of time, normally up to 30 minutes, so steps 1 and 2 may be followed by multiple 3 and 4 pairs until the SAML token expires.

B.3 Usage Scenarios

B.3.1 Independent Identity Provider (STS)



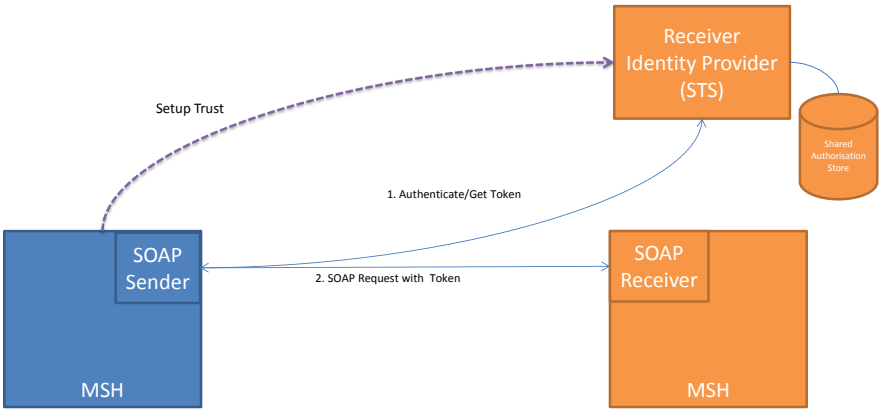
In this scenario, the Identity Provider is an independent third party. They have means of establishing the Sender's identity, obtaining the required attributes and preparing a SAML token that can be used with the Receiver.

The Receiver has established a one off setup with the Identity Provider, exchanging credentials so that the Receiver can validate tokens from the Identity Provider and the Identity Provider can encrypt verification privately for the receiver that may be included in the SAML token.

B.3.1.1 Key Benefits

- The Identity Provider handles all aspects of identifying the user including issues around credential loss or rollover.
- The Receiver only needs to accept one type of credential, the SAML token. The Sender can authenticate with any type of credential supported by the Identity Provider provided the required claims are available.

B.3.2 Receiver End Authorization Provider (STS)



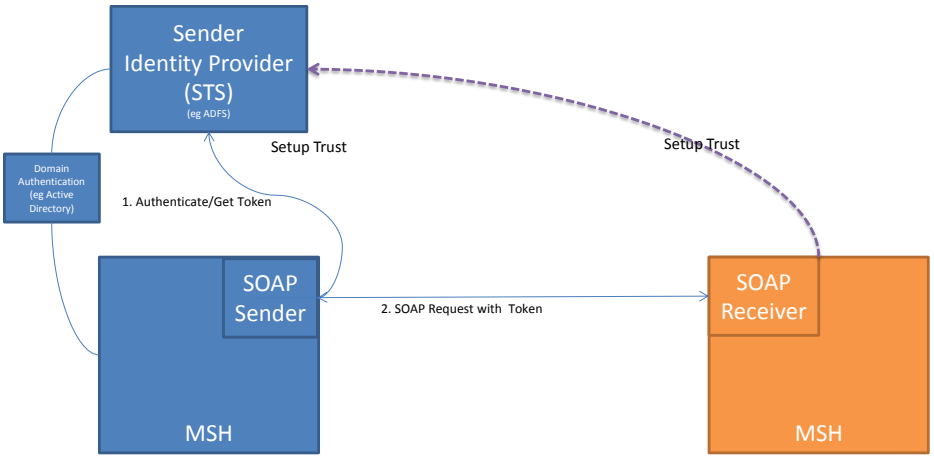
The Identity Provider can live in the receiver's domain. Typically this approach is employed when there is a need to share authorization information across systems and/or technologies.

The Receiver Identity Provider could provide identity and authorization information for a customer portal as well as eBusiness Messaging, providing a single point for storage and administration of the information.

B.3.2.1 Key Benefits

- Identity and Role information are available in web service context without access to external systems or requiring external systems to populate a repository in the MSH.
- Role information is maintained separately and is accessible and consistent across channels.

B.3.3 6 Sender End Authorization Provider (STS)



In business models where the Sender is in a desktop application rather than a back end system, individual users in a business will have various authorities to represent that business.

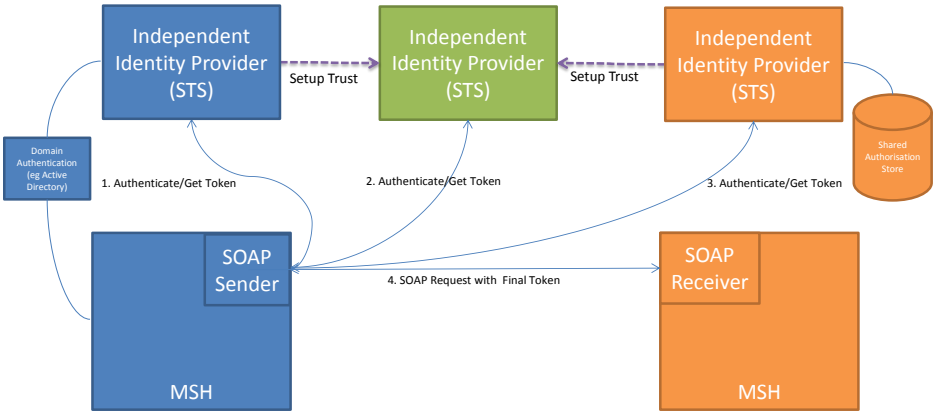
In this case, the Sender will contact an Identity Provider inside its own business for a SAML token. This will be transparent to the user, but will be performed by the domain authentication system. The issued SAML token will be created by the STS with the appropriate claims to indicate the user's authority to represent the business.

B.3.3.1 Key Benefits

- No additional credentials for the users to keep (usernames or certificates) in order transact with external parties.
- Business manages its own authorizations rather than having to rely on external services to stop its users exceeding their authority.

B.3.4 Chained Identity Provider Model

Identity Providers can be chained. That is, one SAML token can be used to obtain another.



In this model, the Sender obtains a series of tokens:

Their domain logon gets them a token that flags them as a valid organization representative with a set of established roles through its issued SAML token.

The organization SAML token is presented to the Independent Identity Provider which establishes the organizations identity and passes through claims that can be legitimately asserted by the organization as well as enriching the SAML token with any required claims that it may have.

The Receiver Identity Provider then issues a SAML token based on the information provided by the Independent Identity Provider, enriching it as appropriate with information that it holds about the business and/or its representative.

B.3.4.1 Key Benefits

- Organizations can control what external roles their representatives are entitled to.
- Sending organizations only need to be registered with one Identity Provider and not with each Receiver.
- Receivers can override or augment identity/role information in a single place before it enters internal systems.
- Identity information can be shared between systems using a standards based mechanism.

Appendix C. Revision History

Revision	Date	Editor	Changes Made
1.0	17-July-13	Ian Otto	Initial Version
1.1	12-August-13	Ian Otto	Incorporated initial review comments
1.2	20-August-13	Ian Otto	Second round review comments, additional explanation and diagrams on WS-Trust step, minor wording changes, changes to authorization PMode.
1.3	28-August-13	Ian Otto	Updated the example with Dale Moberg's comments on which certificates were whose.
1.4	2-Sept-13	Ian Otto	Incorporate comments from Pim van der Eijk around token lifetimes and miscellaneous small corrections

