WS-Federation: Active Requestor Profile

Version 1.0
July 8, 2003

Authors

Siddharth Bajaj, VeriSign
Giovanni Della-Libera, Microsoft
Brendan Dixon, Microsoft
Maryann Hondo, IBM
Matt Hur, Microsoft
Chris Kaler (Editor), Microsoft
Hal Lockhart, BEA
Hiroshi Maruyama, IBM
Anthony Nadalin (Editor), IBM
Nataraj Nagaratnam, IBM
Andrew Nash, RSA Security
Hemma Prafullchandra, VeriSign
John Shewchuk, Microsoft

Copyright Notice

Abstract

This specification defines how the cross trust realm identity, authentication and authorization federation mechanisms defined in WS-Federation are used by active requestors such as SOAP-enabled applications.

Modular Architecture

By using the XML, SOAP and WSDL extensibility models, the WS* specifications are designed to be composed with each other to provide a rich Web services environment. WS-Federation: Active Requestor by itself does not provide a complete security solution for Web services. WS-Federation: Active Requestor is a building block that is used in conjunction with other Web service and application-specific protocols to accommodate a wide variety of security models.

Status

This WS-Federation Active Requestor Specification is an initial public draft release and is provided for review and evaluation only. BEA, IBM, Microsoft, RSA Security and VeriSign hope to solicit your contributions and suggestions in the near future. BEA, IBM, Microsoft, RSA Security and VeriSign make no warrantees or representations regarding the specifications in any manner whatsoever

Table of Contents

1. Introduction

1.1. Goals and Requirements

1.1.1 Requirements

1.1.2. Non-Goals

1.2. Notational Conventions

1.3. Namespaces

1.5. Terminology

2. Model

3.1. Sign-On

3.2. Sign-Out

3.3. Attributes
3.4. Pseudonyms
4. Syntax
4.2. Requesting Security Tokens
4.3. Returning Security Tokens
4.4. Sign-Out Syntax
4.5. Attribute Requests
4.6. Pseudonym Requests
5. Detailed Example
6. Additional Examples
6.1. No Resource STS
6.2. 3rd-Party STS
6.3. Delegated Resource Access
7. Security Tokens
7.1. X.509v3
7.2. Kerberos
7.3. XrML
7.4. SAML
8. Error Handling
9. Security Considerations
10. Acknowledgements
11. References

1. Introduction

The WS-Federation specification defines an integrated model for federating identity, authentication and authorization across different trust realmss. This specification defines how the federation model is applied to active requestors such as SOAP applications.

1.1. Goals and Requirements

The primary goal of this specification is to define mechanisms for federation of identity, authentication, and authorization information as applied to active requestors.

1.1.1 Requirements

The following list identifies the key driving requirements for this specification:

- Enable sharing of identity, authentication, and authorization data between and through active requestors
- Brokering of trust and security token exchange in a active requestor environment

- Optional hiding or protection of identity information and other attributes in a active requestor environment

1.1.2. Non-Goals

The following topics are outside the scope of this document:

- Definition of message security or trust establishment/verification protocols
- Specification of new security token formats

1.2. Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

When describing abstract data models, this specification uses the notational convention used by the XML Infoset. Specifically, abstract property names always appear in square brackets (e.g., [some property]).

When describing concrete XML schemas, this specification uses the notational convention of WS-Security. Specifically, each member of an element's [children] or [attributes] property is described using an XPath-like notation (e.g., /x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element wildcard (<xs:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xs:anyAttribute/>).

1.3. Namespaces

The following namespaces are used in this document:

Prefix	Namespace
S	http://www.w3.org/2002/06/soap-envelope
wsse	http://schemas.xmlsoap.org/ws/2003/07/secext
wsu	http://schemas.xmlsoap.org/ws/2002/07/utility
wp	http://schemas.xmlsoap.org/ws/2002/12/policy

1.4. Terminology

The following definitions outline the terminology and usage in this specification.

Active Requestor – An *active requestor* in a Federation is an application (possibly a Web browser) that is capable of issuing (and receiving) SOAP messages such as those described in WS-Security and WS-Trust.

Claim – A *claim* is a declaration made by an entity (e.g. name, identity, key, group, privilege, capability, attribute, etc).

Security Token – A *security token* represents a collection of claims.

Signed Security Token – A *signed security token* is a security token that is asserted and cryptographically signed by a specific authority (e.g. an X.509 certificate or a Kerberos ticket)

Proof-of-Possession Token – A *proof-of-possession token* is a security token that contains data that a sending party can use to demonstrate proof-of-possession. Typically, although not exclusively, the proof-of-possession information is encrypted with a key known only to the sender and recipient parties.

Digest – A *digest* is a cryptographic checksum of an octet stream.

Signature - A *signature* is a value computed with a cryptographic algorithm and bound to data in such a way that intended recipients of the data can use the signature to verify that the data has not been altered since it was signed by the signer.

Security Token Service (STS) - A *security token service* is a Web service that issues security tokens (see WS-Security and WS-Trust). That is, an STS makes claims based on evidence, to entities that trust the STS. To communicate trust, one service requires proof, such as a security token or set of security tokens, and issues a security token with its own trust statement (note that for some security token formats this can just be a re-issuance or co-signature on the original token). This forms the basis of trust brokering.

Trust - *Trust* is the characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes in a way that is expected.

Trust Domain/Realm - A *Trust DomainRealm* is a security space in which the target of a request can determine whether particular sets of credentials from a source satisfy the relevant security policies of the target. The target may defer trust to a third party thus including the trusted third party in the Trust Realm.

Direct Trust – *Direct trust* is when a relying party accepts as true all (or some subset of) the claims in the token sent by the requestor.

Direct Brokered Trust – *Direct Brokered Trust* is when one party trusts a second party who, in turn, trusts or vouches for, a third party.

Indirect Brokered Trust – *Indirect Brokered Trust* is a variation on direct brokered trust where the second party negotiates with the third party, or additional parties, to assess the trust of the third party.

Signature validation – *Signature validation* is the process of verifying that the message received is the same as the one sent.

Sender Authentication – *Sender authentication* is corroborated authentication among Web service actors/roles indicating the sender of a Web service message (and its associated data). Note that it is possible that a message may nave multiple senders if authenticated intermediaries exist. Also note that it is application-dependent (and out of scope) as to how it is determined who first created the messages as the message originator might be independent of, or hidden behind an authenticated sender.

Realm or Domain – A *realm* or *domain* represents a single unit of security administration or trust.

Federation – A *federation* is a trusted relationship established by a collection (at least two) of realms. The level of trust may vary, but typically includes authentication and may include authorization.

Identity Provider – *Identity Provider* is an entity that acts as a peer entity authentication service to end requestors and data origin authentication service to service providers (this is typically an extension of a security token service)

Single Sign On (SSO) – *Single Sign On* is an optimization of the authentication sequence to remove the burden of repeating actions placed on the end requestor. To facilitate SSO, an element called an Identity Provider can act as a proxy on a requestor's behalf to provide evidence of authentication events to 3rd parties requesting information about the requestor. These Identity Providers are trusted 3rd parties and need to be trusted both by the requestor (to maintain the requestor's identity information as the loss of this information can result in the compromise of the requestors identity) and the Web services which may grant access to valuable resources and information based upon the integrity of the identity information provided by the IP.

Identity Mapping – *Identity Mapping* is a method of creating relationships between identity properties. Some Identity Providers may make use of id mapping.

Sign-Out – A *sign-out* is the process by which a principal indicates that they will no longer be using their token and services in the realm can destroy their token caches for the principal.

2. Model

The WS-Federation specification defines a model and set of messages for brokering trust and federating identity and authentication information across different trust realms. This chapter presents how this model is applied to active requestors such as Web services requestors.

The federation model described in WS-Federation builds on the foundation established by WS-Security and WS-Trust. Consequently, this profile defines mechanisms for requesting, exchanging, and issuing security tokens within the context of active requestors.

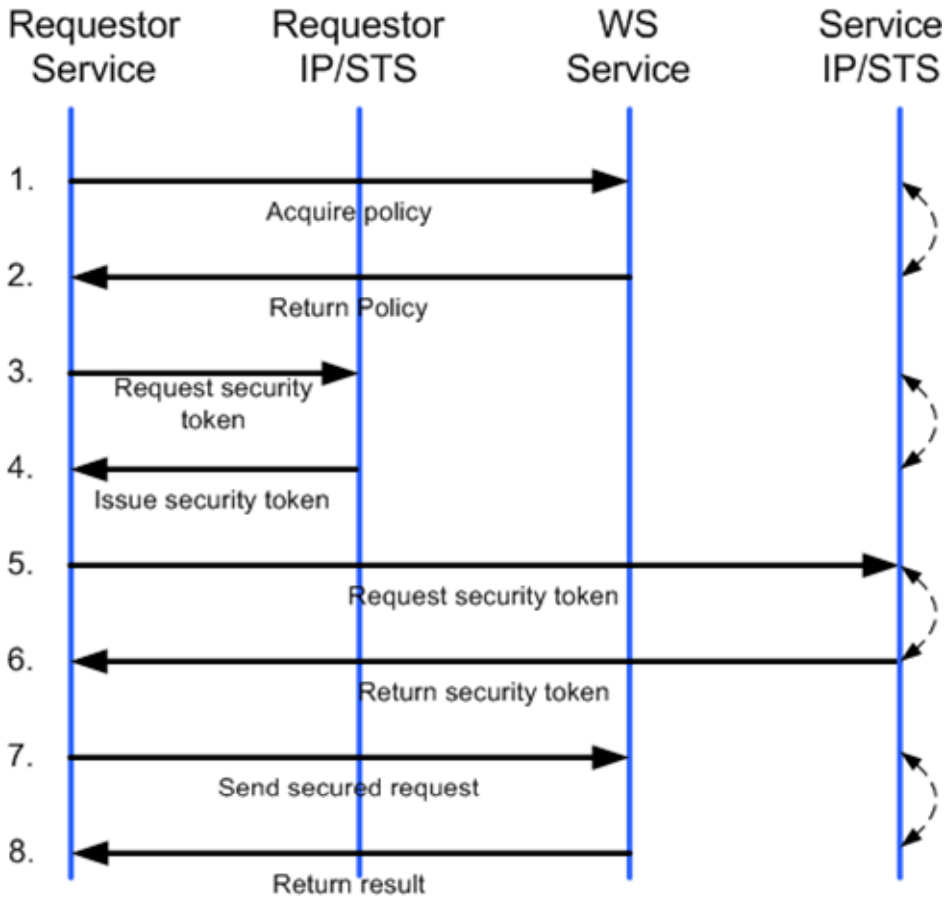
The model defined in this specification allows for support of different but compatible message exchanges. For example, the resource may act as its own security token service (STS) and does not use a separate service (or even URI) thereby eliminating some steps. It is expected that subsequent profiles will be defined to characterize specific exchange patterns.

3.1. Single Sign On

Since *active requestors* are capable of issuing their own messages, they can make use of the mechanisms defined within WS-Security, WS-Trust, and WS-Federation.

At a high-level, policy is used to indicate communication requirements. Requestors can obtain the policy ahead of time or via error responses from services. In general,

requestors are required to obtain a security token (or tokens) from their Identity Provider (or STS) when they authenticate themselves. The IP/STS generates a security token for use by the federated party. This is done using the mechanisms defined in WS-Trust. In some scenarios, the target service acts as its own IP/STS so communication with an additional service isn't required. Otherwise the requestor may be required to obtain additional security tokens from service-specific or service-required identity providers or security token services. The figure below illustrates one possible flow.



While the example above doesn't illustrate this, it is possible that the WS-Trust messages for security tokens may involve challenges to the requestors. Refer to WS-Trust for additional information.

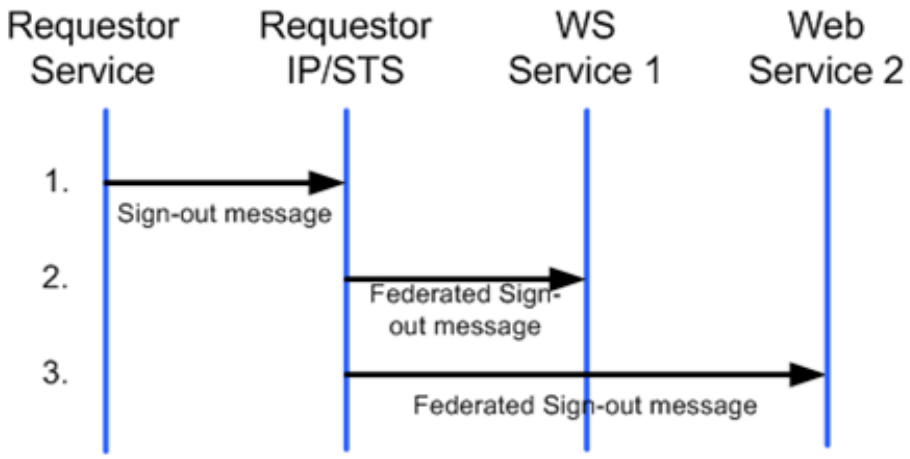
3.2. Sign-Out

Just as it isn't typical for an active requestor to sign-in, it isn't typical to *sign-out* either. However, for those scenarios where this is desirable, the sign-out messages defined in WS-Federation MAY be used.

In situations where federated sign-out messages are desirable, The requestor's IP/STS SHOULD keep track of the realms to which it has issued tokens – specifically the IP/STS for the realms (or resources if different). When the sign-out is received

at the requestor's IP/STS, the requestor's IP/STS is responsible for issuing federated sign-out messages to interested and authorized parties. The exact mechanism by which this occurs is up to the IP/STS, but it is strongly RECOMMENDED that the sign-out messages defined in WS-Federation be used.

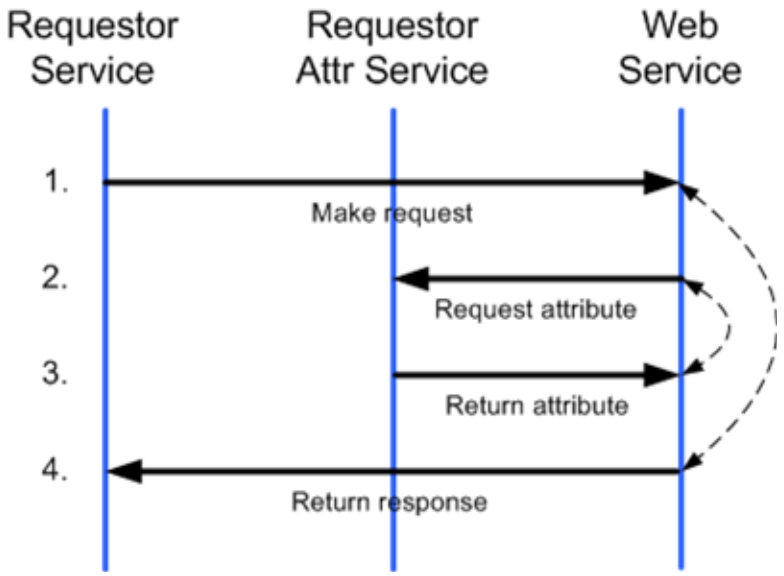
When a federated sign-out message is received at a realm, the realm SHOULD clean-up any cached information and delete any associated state as illustrated in the figure below:



3.3. Attributes

For active requestors, attribute services are identified via WS-Policy as described in WS-Federation. Web services and other authorized parties can obtain or even update attributes using the messages defined by the specific attribute service.

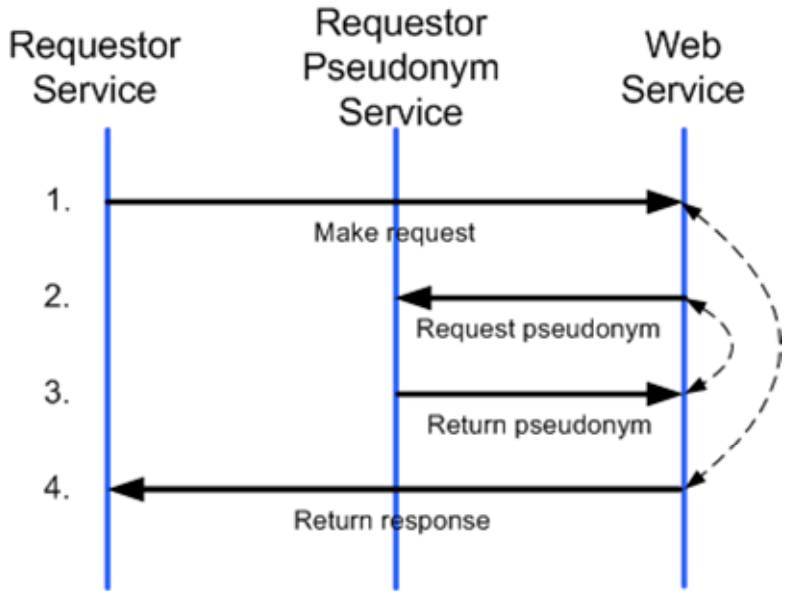
The figure below illustrates a scenario where a requestor issues a request to a Web service. The request may include the requestor's policy or it may be already cached at the service or the requestor may use WS-PolicyExchange. The Web service issues a request to the requestor's attribute service to obtain the values of a few attributes, WS-Policy may be used to describe the location of the attribute service. The service is authorized so the attributes are returned. The request is processed and a response is returned to the requestor.



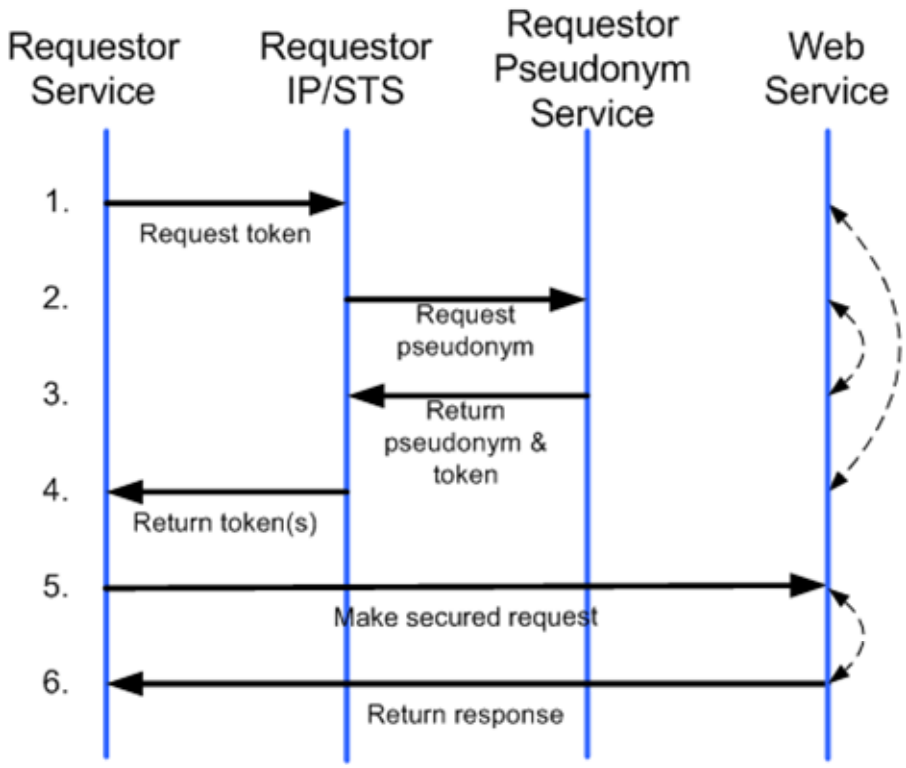
3.4. Pseudonyms

For active requestors, pseudonym services are identified via WS-Policy as described in WS-Federation. Services and other authorized parties can obtain or manage pseudonyms using the messages defined in WS-Federation.

The figure below illustrates a scenario where a requestor issues a request to a Web service. The request may include the requestor's policy and the location of the requestor's pseudonym service or it may be already cached at the Web service. The Web service issues a request to the requestor's pseudonyms service to obtain the pseudonyms that are authorized by the security token. The Web service is authorized so the pseudonym is returned. The request is processed and a response is returned to the requestor.



As described in WS-Federation, the pseudonym and IP/STS can interact as part of the token issuance process. The figure below illustrates a scenario where a requestor has previously associated a pseudonym and associated security token for a specific realm. When the requestor requests a security token to the domain/realm, the pseudonym and token are obtained and returned to the requestor. The requestor uses these security tokens for accessing the Web service.



4. Syntax

This section defines the syntax for the federation mechanisms described in the model above.

4.1. Requesting Security Tokens

Security tokens are requested using the `<RequestSecurityToken>` message defined in the WS-Trust specification.

4.2. Returning Security Tokens

Security tokens are returned using the `<RequestSecurityTokenResponse>` message defined in the WS-Trust specification.

4.3. Sign-Out Syntax

Explicit sign-out notification is performed using the `<SignOut>` message defined in the WS-Federation specification.

Similarly, federated sign-out messages use the same message element.

4.4. Attribute Requests

Attributes are requested and updated using messages specific to the attribute services as described in the WS-Federation Specification. This specification doesn't mandate a specific attribute store technology.

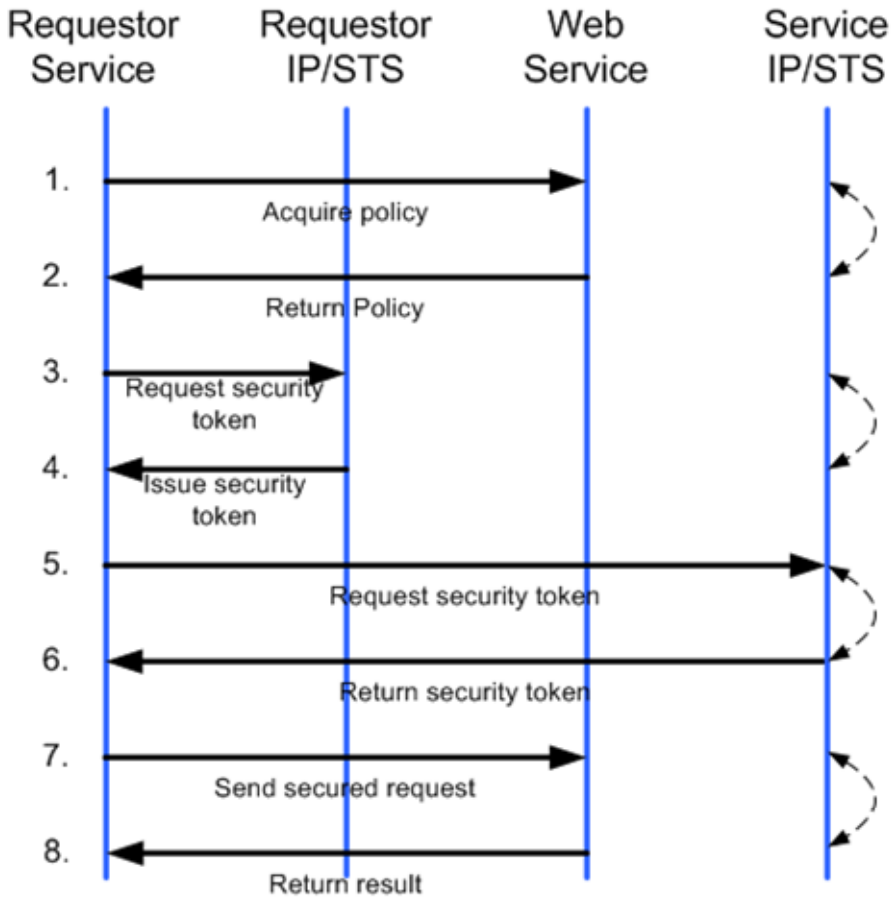
4.5. Pseudonym Requests

Pseudonyms are requested and updated using the messages and mechanisms described in the WS-Federation specification.

5. Detailed Example

This section provides a detailed example of the protocol defined in this specification. The exact flow can vary significantly; however, the following diagram and description depict a *common* sequence of events.

In this scenario, an active requestor is attempting to access a service which requires security authentication to be validated by the resource's security token service.



Step 1: Acquire Policy

If the requestor doesn't already have the policy for the service, it can obtain it using the mechanisms defined in WS-MetadataExchange.

Step 2: Return Policy

The requested policy is returned using the mechanisms defined in WS-MetadataExchange.

Step 3: Request Security Token

The requestor requests a security token from its IP/STS (assuming short-lived security tokens) using the mechanisms defined in WS-Trust (`<RequestSecurityToken>`)

Step 4: Issue Security Token

The IP/STS returns a security token (and optional proof of possession information) using the mechanisms defined in WS-Trust (`<RequestSecurityTokenResponse>` and `<RequestedProofToken>`)

Step 5: Request Security Token

The requestor requests a security token from the Web services IP/STS for the target Web service using the mechanisms defined in WS-Trust (`<RequestSecurityToken>`). Note that this is determined via policy or some out-of-band mechanism.

Step 6: Issue Security Token

The Web service's IP/STS returns a token (and optionally proof of possession information) using the mechanisms defined in WS-Trust (`<RequestSecurityTokenResponse>`)

Step 7: Send secured request

The requestor sends the request to the service attaching and securing the message using the issued tokens as described in WS-Security.

Step 8: Return result

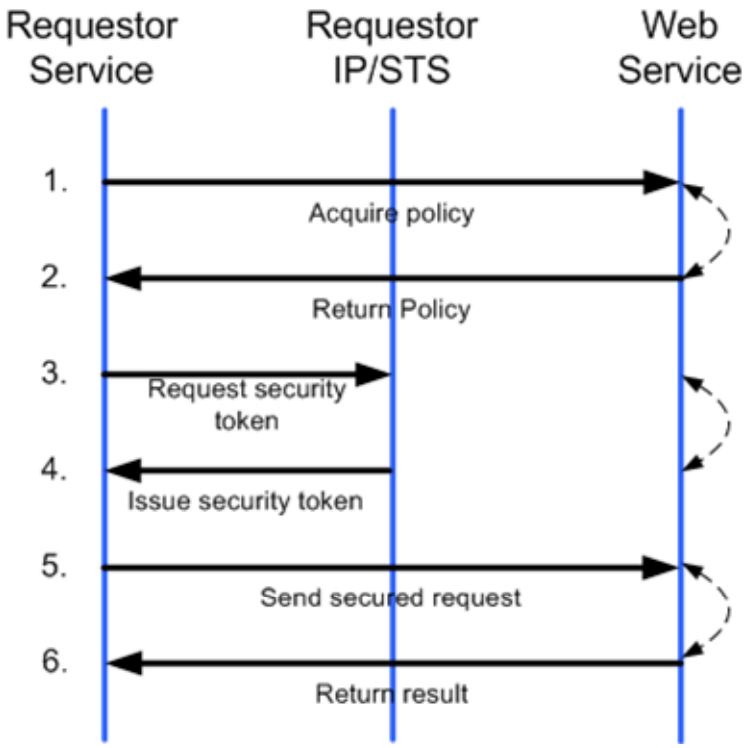
The service issues a secured reply using its security token.

6. Additional Examples

This section presents interaction diagrams for additional active requestor scenarios.

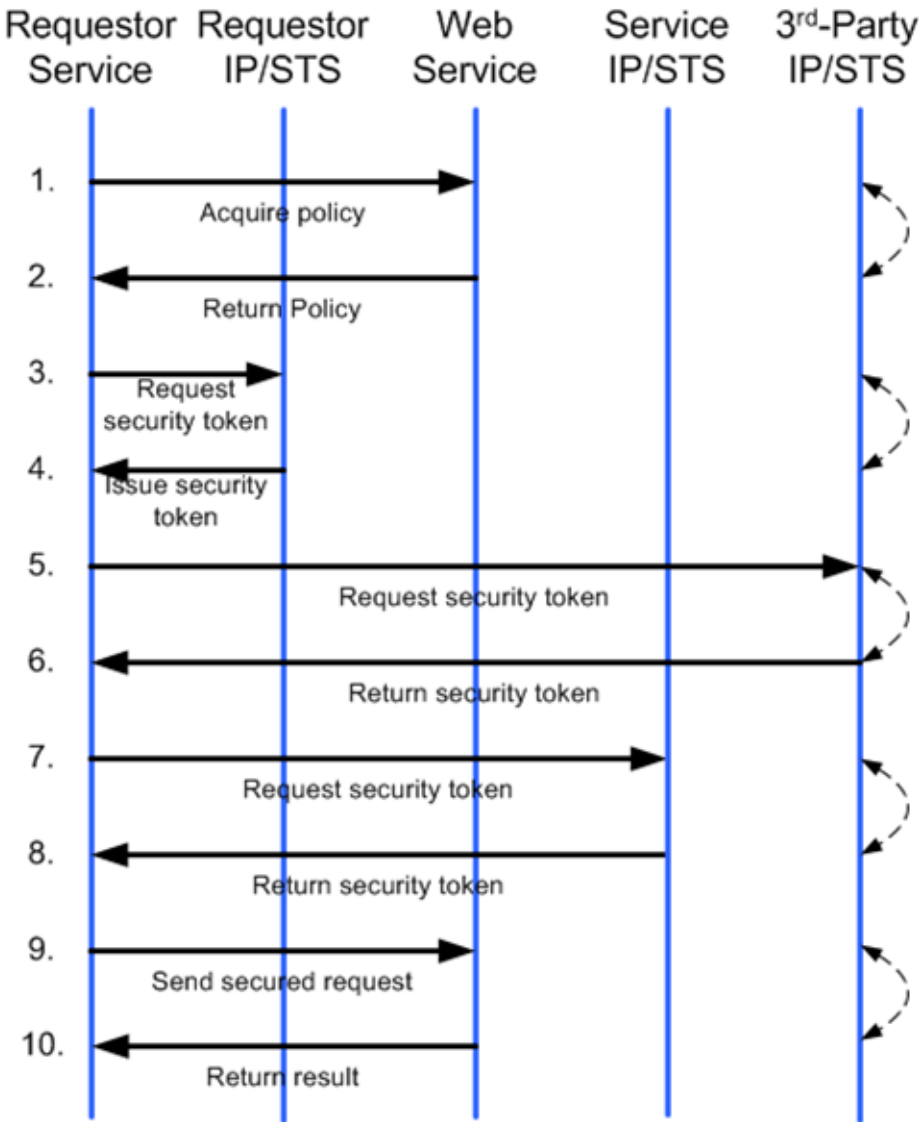
6.1. No Resource STS

The figure below illustrates the resource access scenario above, but without a resource STS. That is, the Web service acts as its own STS:



6.2. 3rd-Party STS

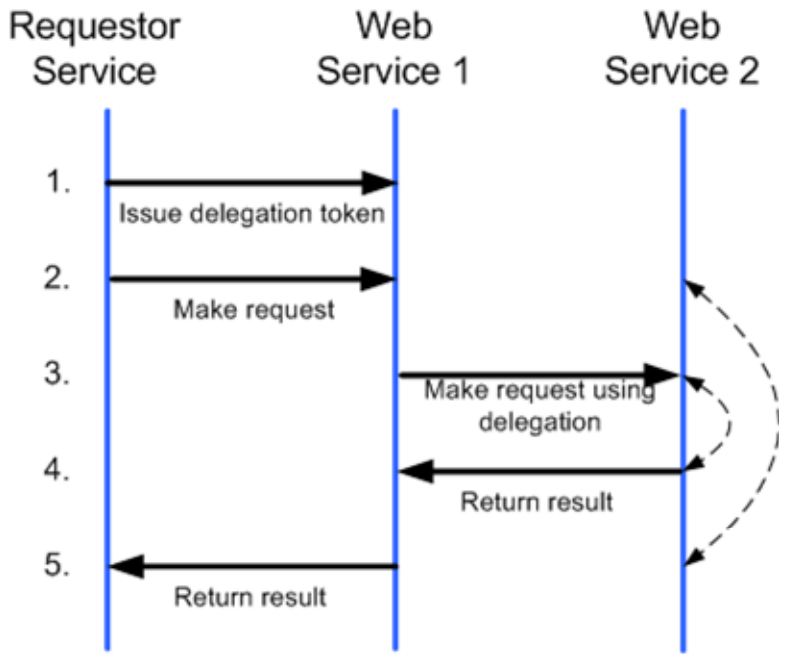
The figure below illustrates the resource access scenario above, but trust is brokered through a 3rd-party STS:



Note that 3rd-Party IP/STS is determined via policy or some out-of-band mechanism.

6.3. Delegated Resource Access

The figure below illustrates where a resource access data from another resource on behalf of the first resource:



In this example, the requestor used a `<RequestSecurityTokenResponse>` as defined in WS-Trust to issue the delegation token in Step 1. This provides to Web Service 1 the necessary information so that Web Service 1 can act on the requestor's behalf as it contacts Web Service 2.

7. Security Tokens

When accepting security tokens, recipients SHOULD:

- Verify the token is formatted correctly
- Verify STS signature
- Verify the token validity interval
- Verify properties requested by policy such as required authentication type, maximum time since authentication instant (e.g. a password must have been submitted within 1 hour), identity properties etc.

This chapter describes token format-specific requirements but it does not mandate usage of a particular token type.

7.1. X.509v3

This specification places the following requirements on X.509 tokens:

- Tokens MUST contain the name of the issuing authority and a signature of the issuing authority over the whole token unless a secure channel is used to communicate the token. That is, a signature element over the assertions. Note that it is RECOMMENDED that a signature be used even if a secure channel is used.
- Tokens MUST contain the subject identifier uniquely identifying the subject for whom the token was granted. X.509 does not specify rules for `Principal`

field. X.509 tokens conformant with this specification SHOULD assure the principals issued are unique across realms and also the realm SHOULD be derivable from the principal name.

- Tokens MAY contain the time of initial authentication, validity interval and the type of authentication that was performed.
- Tokens MAY contains Certificate Revocation Information, such as a CRL distribution point
- X.509 certificates MUST be carried within a `wsse:BinarySecurityToken` element whose ValueType is `wsse:X509v3`.

7.2. Kerberos

This specification places the following requirements on Kerberos tokens:

- Kerberos ticket-granting tickets MUST be carried within a `wsse:BinarySecurityToken` element whose ValueType is `wsse:Kerberosv5TGT`.
- Kerberos service tickets MUST be carried within a `wsse:BinarySecurityToken` element whose ValueType is `wsse:Kerberosv5ST`.
- The symmetric key used SHOULD be derived from the desired realm

7.3. XrML

This specification places the following requirements on XrML tokens:

- Processors that MUST support the `xrml:issuer` element with and without contained signatures. Processors SHOULD NOT include a contained signature unless the `xrml:license` conveys the key (directly or indirectly).
- Tokens that contain signatures in one or more `xrml:issuer` elements MUST declare all XML namespaces on the `xrml:license` element.
- Processors MUST include an `xrml:issuer` element identifying the issuer under `xrml:details`.
- Processors MUST include within the `xrml:issuer` element an `xrml:validityInterval` when the `xrml:license` token conveys the key (directly or indirectly). The `xrml:validityInterval` MUST contain both `xrml:notBefore` and `xrml:notAfter` elements.
- Tokens SHOULD contain a recipient identifier indicating the scope of usage (such as the resource or realm) - this is represented by `grant resource`, with the tacit assumption that the realm is used.

7.4. SAML

This specification places the following requirements for SAML tokens:

- Tokens MUST contain a signature of the issuing authority over the whole token unless a secure channel is used to communicate the token. That is, a signature element over the SAML assertion. Note that it is RECOMMENDED that a signature be used even if a secure channel is used.

- Tokens MUST contain the subject identifier uniquely identifying the subject for whom the token was granted. SAML does not specify rules for `NameIdentifier` element. The SAML assertions conformant with this specification SHOULD assure the identifiers issued are unique across realms and also the realm SHOULD be derivable from the subject identifier.
- Tokens SHOULD contain a recipient identifier indicating the scope of usage (such as the resource or realm) - the `AudienceRestriction` or `Recipient` elements in the SAML assertion.
- Tokens MUST contain the time of initial authentication, validity interval and the type of authentication that was performed. The validity interval in the SAML assertion is satisfied by the `NotBefore` and `NotOnOrAfter` attributes of the Conditions element. The initial authentication type and time are covered by the attributes of `AuthenticationStatement` element.
- Tokens MAY contain additional identity information. If they do, the schema describing the additional information MUST be understood by the recipient or the token MUST be rejected.

8. Error Handling

Errors are handled using the mechanisms described in the WS-Security, WS-Trust, WS-Federation, and any referenced specifications. No additional error semantics or error codes are defined by this specification.

9. Security Considerations

This section outlines security considerations beyond those identified in WS-Federation and other Web service security specifications.

If a security token is not self-securing, it SHOULD be included in some form of message integrity mechanism such as the mechanisms described in WS-Security.

If privacy is a concern, the security tokens MAY be encrypted for the authorized recipient(s) using the mechanisms described in WS-Security.

10. Acknowledgements

This specification has been developed as a result of joint work with many individuals and teams, including:

Tim Hahn, IBM
Heather Hinton, IBM
Bronislav Kavsan, RSA Security
Anthony Moran, IBM
Robert Philpott, RSA Security
Yordan Rouskov, Microsoft
Shane Weeden, IBM
Jeff Spelman, Microsoft

11. References

[KEYWORDS]

S. Bradner, "Key Words for Use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997.

[SOAP]
W3C Note, "SOAP: Simple Object Access Protocol 1.1," 08 May 2000.
Draft, SOAP 1.2, http://www.w3.org/TR/soap12-part0/
Draft, SOAP 1.2, http://www.w3.org/TR/soap12-part1/
Draft, SOAP 1.2, http://www.w3.org/TR/soap12-part2/

[URI]
T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.

[WS-Federation]
"Web Services Federation Language", BEA, IBM, Microsoft, RSA Security, VeriSign, July 2003

[WS-Security]
"Web Services Security Language", IBM, Microsoft, VeriSign, April 2002.
"WS-Security Addendum", IBM, Microsoft, VeriSign, August 2002.
"WS-Security XML Tokens", IBM, Microsoft, VeriSign, August 2002

[WS-Policy]
"Web Services Policy Framework", BEA, IBM, Microsoft, SAP, December 2002

[WS-PolicyAttachment]
"Web Services Policy Attachment Language", BEA, IBM, and Microsoft, SAP, December 2002

[WS-PolicyAssertions]
"Web Services Policy Assertions Language", BEA, IBM, Microsoft, SAP, December 2002

[WS-Trust]
"Web Services Trust Language", IBM, Microsoft, RSA, VeriSign, December 2002

[WS-SecureConversation]
"Web Services Secure Conversation Language", IBM, Microsoft, RSA, VeriSign, December 2002

[WS-SecurityAssertions]
"Web Services Security Assertions Language", IBM, Microsoft, RSA, Verisign December 2002

[XML-ns]
W3C Recommendation, "Namespaces in XML," 14 January 1999.