Understanding WS-Federation

May 28, 2007

Version 1.0

Authors

Marc Goodner, Microsoft Corporation
Maryann Hondo, IBM
Anthony Nadalin, IBM
Michael McIntosh, IBM
Don Schmidt, Microsoft Corporation

Copyright Notice

Summary

This paper is intended to help the reader understand the features of WS-Federation by describing the use of the specification in selected application scenarios.

WS-Federation extends WS-Trust to provide a flexible Federated Identity architecture with clean separation between trust mechanisms, security token formats, and the protocol for obtaining tokens.

This architecture enables a reusable security token service model and protocol to address the identity requirements of both web applications and web services in a variety of trust relationships.

The features of WS-Federation can be used directly by SOAP clients and web services. WS-Federation also defines syntax for expressing the WS-Trust protocol and WS-Federation extensions in a browser based environment. The intention of this functionality is to provide a common model for performing Federated Identity operations for both web services and browser-based applications.

This paper briefly reviews WS-Trust and then describes how WS-Federation builds upon the Security Token Service model defined in that standard. A basic overview of the features of WS-Federation is provided. Key concepts are explained by examples of usage in application scenarios.

This paper is intended for an audience familiar with web services security topics. Familiarity with other web service specifications (i.e. WS-Trust and WS-Federation) is helpful but not essential.

Contents

1. Introduction

WS-Security, WS-Trust, and WS-SecurityPolicy provide a basic model for federation between Identity Providers and Relying Parties. These specifications define mechanisms for codifying claims (assertions) about a requestor as security tokens which can be used to protect and authorize web services requests in accordance with policy. WS-Federation extends this foundation by describing how the claim transformation model inherent in security token exchanges can enable richer trust relationships and advanced federation of services. This enables high value scenarios where authorized access to resources managed in one realm can be provided to security principals whose identities and attributes are managed in other realms. WS-Federation includes mechanisms for brokering of identity, attribute discovery and retrieval, authentication and authorization claims between federation partners, and protecting the privacy of these claims across organizational boundaries. These mechanisms are defined as extensions to the Security Token Service (STS) model defined in WS-Trust. In addition WS-Federation defines a mapping of these mechanisms, and the WS-Trust token issuance messages, onto HTTP such that WS-Federation can be leveraged within Web browser environments. The intention is to provide a common infrastructure for performing Federated Identity operations for both web services and browser-based applications. A common protocol provides economies with regard to development, testing, deployment and maintenance for vendors and customers alike.

The best way to understand WS-Federation is to see it in use. This paper describes two scenarios that exercise selected features of WS-Federation in different ways. The Enterprise scenario considers bidding on supply-chain contracts; the Healthcare scenario explores providing access to patient records. There is some overlap in the features used. The difference in the application of these features will show the flexibility of the features of the specification. These are advanced scenarios that highlight complex situations with extensive use of WS-Federation. This should not imply that WS-Federation is only useful in situations as complicated as the ones presented here. The goal is to highlight that new scenarios based upon both realistic and complicated situations can be supported while, at the same time, providing a fair amount of coverage of the WS-Federation specification. The goal is not to provide an exhaustive exercise of the features of WS-Federation. For the purposes of this paper, we assert that the basic concepts of the web (passive) requestor, where web service messages are mapped into browser HTTP messages, are understood. Many businesses have already addressed the simple single sign-on use cases, so this paper does not go into these in detail. The newer features in the specification are introduced with the intention of addressing other web services requestors engaged in business scenarios. More advanced features, such as pseudonyms, are not covered in this paper. Features that will be described in the scenarios that follow are:

- Federation Metadata - Enterprise, Healthcare
- Authorization – Enterprise, Healthcare
- Privacy - Healthcare
- Indicating Specific Policy and Metadata - Enterprise, Healthcare
- Sign out - Healthcare
- Web (passive) Requestors - Enterprise

The organization of the paper is as follows.

Section 2 – WS-Trust and WS-Federation –This section provides a short overview of WS-Trust before describing how WS-Federation builds upon the Security Token Service model defined by WS-Trust. The features of WS-Federation are then summarized.

Section 3 – Enterprise Scenario – The section presents an Enterprise scenario centered on one organization offering a Request for Proposal (RFP) via an online service and another organization accessing this service and placing bids. This scenario demonstrates the use of the following WS-Federation features: Federation Metadata to enable automated service configuration, Application-specific Policy and Metadata to convey additional security semantics, Authorization Context to inform security token service decisions when issuing tokens and Common Claim Types to promote interoperability. This scenario concludes with an example of the Web (passive) Requestor feature to enable the use of a web browser client.

Section 4 – Healthcare Scenario – This section presents a Healthcare scenario centered on patient record access in an emergency situation. This scenario demonstrates the use of the following WS-Federation features: Federation Metadata, Application-specific Policy and Metadata, Authorization Context and Common Claim Types. This example presents different uses of these features from the previous Enterprise scenario. The Healthcare scenario also demonstrates the use of Privacy Protection by a requestor to indicate that a sensitive claim which may be issued in a security token should be protected by encryption as well as the use of federated Sign Out.

Appendix A - PRIP Claim Types – This section documents claim types that were defined in the Passive Requestor Interoperability Profile used in WS-Federation interoperability testing. These claim types are used in some of the examples in the scenarios described above. In addition to showing the original mapping of the claim types to SAML 1.1 assertions, this section illustrates mappings to the WS-Federation metadata document UriNamedClaimTypesOffered element and the Authorization Common Claims dialect.

Appendix B – Acknowledgements – This acknowledges the inputs and support provided by other individuals in the creation of this paper.

Appendix C – XML Namespaces – This section lists all the XML Namespaces that are used in this document.

This is a non-normative document and does not provide a definitive specification of the WS-Federation Policy Language.

Appendix D – References – This section provides a list of references which readers may find helpful for more detailed information.

2. WS-Trust and WS-Federation

2.1.WS-Trust

The mechanisms defined in WS-Security, WS-Trust, and WS-SecurityPolicy provide the basic model for federation. Trust is the bedrock of any security model. To begin, we define, what is "trust"? Trust is the expression between parties that one party to a relationship will believe statements (claims) made by another party; it is based on evidence – history, experience, documents, etc. – and personal risk tolerance. Trust in a person can be based on past experience where one party expresses expectations about future behavior of another party based on the shared experience. Some examples are:

- I trust that buying a specific book from Alice is safe because I have bought many books from Alice over the past 10 years.
- I do not trust that buying a specific book from Bob is safe because I have never bought books from Bob before.

In general trust in information is substantiated by claims about the origin, correctness and usage rights of the information. Specifically with respect to web services trust principles, this often refers to the likelihood that the recipient of a message with a security token will believe that the claims the token makes about a subject are true because the recipient recognized the origin of the token and has experience with the issuer of the token following good practices.

WS-Security provides a core function by defining mechanisms for assuring message authenticity, integrity and confidentiality through the use of security tokens. WS-SecurityPolicy enables the description of the security requirements of services via assertions about the security mechanisms of the services (i.e. algorithms and types of tokens that the service accepts). Using these assertions web services are able to recognize and assess the types of security tokens and claims that are required for exchanging messages securely. WS-Trust provides an additional piece of the foundation for federation by defining a service model, the Security Token Service (STS), and a protocol for requesting/issuing these security tokens which are used by WS-Security and described by WS-SecurityPolicy.

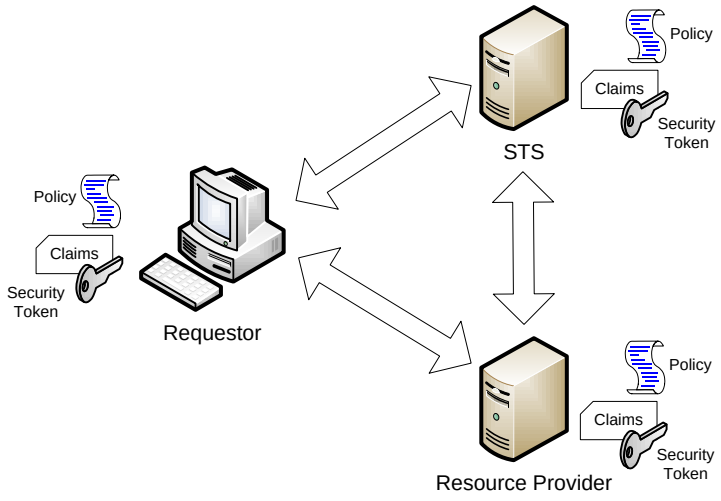

Figure 1 – Security Token Service (STS) Model

Figure 1 illustrates a view of the Security Token Service (STS) model defined by WS-Trust. Each arrow represents a possible communication path between the participants. Each participant has its own policies which combine to determine the security tokens and associated claims required to communicate along a particular path. From the Requestor's perspective the communication flow starts with the identification of a web service, or Resource Provider, that the requestor wishes to access. The requestor queries the Resource Provider for its policies to determine security requirements. Using WS-SecurityPolicy expressions the Requestor can check its own capabilities to determine if requestor has a security token that meets the requirements to access the Resource Provider. If the requestor does not have an acceptable token it might be able to request one from an appropriate STS which can also be identified in the Resource Provider's policy. Each STS has its own associated policy and being a web service, the Requestor can query the STS to determine the security requirements for requesting a particular type of token for use.

Figure 1 depicts the STS functioning in the role of an Identity Provider (IP). The primary function of an STS in this role is to issue identity tokens that contain claims about a security principal that correspond to the Requestor. An IP STS can also be used by a Resource Provider to validate tokens it has received from Requestors. A Resource Provider is frequently referred to as a Relying Party to indicate that it relies upon tokens issued by an STS to grant/deny access to the resources it controls. For the purposes of this document the terms Resource Provider and Relying Party are synonymous, and may be used interchangeably. The difference is simply one of perspective. Typically Resource Provider is used when referring to this participant from the perspective of the Requestor, and Relying Party is used when referring to it from the perspective of the STS who is identifying the ultimate recipient of the token. Therefore the abbreviation RP refers to either concept through the rest of this paper.

WS-Trust introduces protocol mechanisms independent of any particular application for requesting, issuing, renewing, cancelling and validating security tokens which can be exchanged to authenticate

principals and protect resources. The core of this protocol is the request-response message pair, Request Security Token (RST) and Request Security Token Response (RSTR).

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
    </wsa:Action>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body>
    <wst:RequestSecurityToken>
      <wst:TokenType>
        http://example.org/mySpecialToken
      </wst:TokenType>
      <wst:RequestType>
        http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
      </wst:RequestType>
    </wst:RequestSecurityToken>
  </s:Body>
</s:Envelope>
```

Example Request Security Token (RST) message

The above fragment is a simple example of a Request Security Token (RST) message, illustrating the token type being requested (mySpecialToken) and the request type in the body of the message (Issue). Other request types are possible, for example: Cancel, Renew and Validate. WS-Trust is not limited to any particular token type. This example illustrates a custom token type being requested. By remaining agnostic of the type of token being transmitted WS-Trust enhances the interoperability of the basic protocol and allows migration of customer deployed products as the industry introduces new and improved security token formats.

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/Issue
    </wsa:Action>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body>
    <wst:RequestSecurityTokenResponseCollection>
      <wst:RequestSecurityTokenResponse>
        <wst:RequestedSecurityToken>
          <xyz:CustomToken xmlns:xyz="...">
            ...
          </xyz:CustomToken>
        </wst:RequestedSecurityToken>
      </wst:RequestSecurityTokenResponse>
    </wst:RequestSecurityTokenResponseCollection>
  </s:Body>
</s:Envelope>
```

Example Request Security Token Response (RSTR) message

This fragment is an example Request Security Token Response (RSTR) message, illustrating a successful (and simple) issuance of the custom token from the previous RST example. Other responses are possible, including challenges that must be met before the requested token would be issued.

Security Token Services can broker the establishment of a trust relationship between a Resource Provider and other service providers (Identity Providers for example), that are prepared to vouch for the identity, pseudonyms or other attributes which they have associated with a specific principal. In order for a security token from an Identity Provider realm to be useful in a Relying Party realm, the following items are required:

- Trust relationship between the realms
- Agreement upon the syntax and semantics of security tokens
- Endpoints for obtaining policy requirements and requesting security tokens

2.2.WS-Federation

A federation is a collection of realms (security domains) that have established relationships for securely sharing resources. A Resource Provider in one realm can provide authorized access to a resource it manages based on claims about a principal (such as identity or other distinguishing attributes) that are asserted by an Identity Provider (or any Security Token Service) in another realm.

The value of establishing a federation is to facilitate the use of security principal attributes across trust boundaries to establish a federation context for that principal. A Relying Party can then use this context to grant/deny access to a resource. Establishing a federation context when Identity and Resource Providers operate in different realms requires agreement between these parties on what claims are required and frequently requires agreement on mechanisms for securely transporting those claims over unprotected networks. This provides the basis for interoperability. In general it is necessary for participants in a federation to communicate these requirements over a wide variety of trust and communication topologies. Supporting different topologies requires the exchange of metadata describing endpoint references where services may be obtained, plus the potential security policies and communication requirements that must be observed when accessing those endpoints. The exchange of this metadata can be further complicated because the participants in a single federation may have different policies and service providers may participate in multiple federations.

A fundamental goal of WS-Federation is to simplify the development of federated services through cross-realm communication and management of Federation Services by re-using the WS-Trust Security Token Service model and protocol. A variety of Federation Services (e.g. Authentication, Authorization, Attribute and Pseudonym Services) can be developed as variations of the base Security Token Service. Managing, discovering and accessing such services are dramatically simplified when they are all based on a common processing model and speak the same base protocols. Further, reusing an established processing model and having one common protocol reduces the Threat Model variables for implementers and should lead to more robust code.

WS-Federation does not restrict users to a specific security token format. Instead, WS-Federation builds on the WS-Trust encapsulation mechanism, the RST/RSTR, which allows protocol processing to remain

agnostic of the type of token being transmitted. This enhances the interoperability and migration of customer deployed products as the industry introduces new and better security token formats.

Building on the WS-Trust foundation the WS-Federation protocol provides mechanisms that simplify interactions between the participants. A well documented method for exchanging federation metadata makes it easy to bootstrap trust relationships and also to determine policies for obtaining services. Cross organizational identity mapping and distributed sign-out improve the utility and overall security of accessing federated service providers by minimizing the user's need to manage many identifiers and tokens. WS-Federation protocol extensions to WS-Trust allow the enablement of advanced services by integrating pseudonym, attribute and claims-based authorization services with generic Security Token Services seamlessly into the basic security model. Pseudonym services and the claims-based authorization model can be used to further satisfy and enhance user privacy requirements across realm boundaries in a federation. A Resource Provider can describe the set of attributes required to access a resource and an Identity Provider can assert that a particular principal possesses those attributes, without divulging the actual identity of the principal. Finally, within the limitations of standard web browser clients, the security token based capabilities provided by WS-Trust and WS-Federation protocol extensions can be made accessible via HTTP 1.1 mechanisms to be used in browser based scenarios.

The remainder of this section examines each of the above described features of WS-Federation in more detail.

Federation Metadata

Organizations typically decide to establish federated services based on business or legal relationships. After an organization decides to establish a federation, the participants of the federation must publish and exchange configuration information (i.e. federation metadata) that allows them to identify the common services (e.g. token, authorization) to participants in the federation and the policies for accessing them. For web services, this information can be expressed as statements in federation metadata documents and may include supplying WS-Addressing endpoint references (EPRs) to the participants as well as security policies which list the security tokens and claims required to access those end points. In any federation, a mechanism must be established for determining the authenticity of metadata documents. Since a service may be exposed in multiple federations it must be possible to identify the metadata that applies to each distinct context. In addition, it is desirable to help automate this configuration process.

To enable this configuration process, WS-Federation defines a metadata model and document format. The metadata model describes how federation metadata about related services can be discovered and combined. The federation metadata document describes how a service participates and is used within a federation. The metadata defined by WS-Federation augments other metadata about services, (e.g. WSDL and WS-Policy) and is not a replacement for that metadata.

Authorization

An authorization service may be implemented as a special type of Security Token Service which provides decision brokering services for participants in a federation. While the internal processing of an authorization enforcement is implementation specific, interoperability between services in a federation requires a common model for interacting with authorization services. Extensions to RST/RSTR mechanisms, as defined in WS-Trust, may be used for communicating authorization requests and decision outputs.

WS-Federation defines an authorization model that meets these requirements. The protocol also defines two extensions for rich authorization capabilities. The first of these extensions allows additional context about a token request to be passed to an STS in an RST request. The second extension builds on the Claims Dialect mechanism, defined in WS-Trust, to allow Resource Providers and Requestors to indicate specific claims that are required to process requests. Different claim representations are used across different web service implementations to address different application requirements. This sometimes makes it difficult to express claims in a general way across these application implementations. To facilitate interoperability, a simple dialect for expressing common claims in a format-neutral way is defined.

Authentication Types

The WS-Trust specification defines the AuthenticationType parameter to indicate a type of authentication that is required (or performed) or an assurance level with respect to a particular security token request. However, no particular types are defined or required. To facilitate interoperability WS-Federation has identified and defined a set of Universal Resource Identifiers (URIs) for specifying the common authentication types and assurance levels that can be used for the wst:AuthenticationType parameter in RST and RSTR messages.

Attribute Services

The participants in a federation may not always be able to establish a federation context for a principal using only the claims obtained from security tokens. For example, after a principal's original request has been authenticated a Resource Provider may determine that additional information is required to authorize access to advanced functionality. A service provider offers a solution to this common problem by operating an attribute service that requesters may use to obtain this additional information (e.g. Claims). WS-Federation defines a model for either party to access attribute services based upon the security token service concept and reliant on the token issuance protocol defined in WS-Trust.

Pseudonym Services

A pseudonym service is a further specialization of an attribute service which provides alternate identity information for principals who are concerned about the risks of identity fraud. This is enabled transparently by optionally integrating the pseudonym service into the Security Token Service model. It may provide distinct pseudonyms for different scopes, that is, for access to different Resource Providers. A pseudonym service may store tokens associated with a pseudonym to simplify retrieving a pseudonym

and associated tokens in a single security token request. WS-Federation describes how a pseudonym service that is combined with a Security Token Service may map pseudonyms to issued tokens. This includes describing how the mapping may be automatically performed based on the target service for the token. It also defines extensions to the WS-Trust RST/RSTR syntax for requestors to manually specify how pseudonyms should be mapped.

Privacy

Requests made to service providers for security tokens or authorization decisions may include information that is subject to personal or organizational privacy requirements.

WS-Federation defines extensions to the RST/RSTR syntax defined in WS-Trust for a requestor to express its privacy requirements and likewise for a Security Token Service to indicate to the requestor the mechanisms actually used for issuing the token. This includes extensions for indicating parameters that an STS must use if it issues a token, as well as identifying individual sensitive claims in a security token for which the values should be protected by encryption.

WS-Federation also defines a model for how privacy statements can be obtained using mechanisms defined in HTTP, HTTPS, WS-Transfer, WS-ResourceTransfer, WS-Policy or WS-MetadataExchange.

Indicating Specific Policy and Metadata

To enable a dynamic model for federation, it is important to address the case of a requestor attempting to access a target service and finding there may be additional security or contextual requirements based on the specifics of the request. Examples of situations where this may arise are provided in the Enterprise and Healthcare scenarios below. WS-Federation defines a mechanism that allows the target service to indicate to the requestor that additional security semantics apply to a specific request. This enables the requestor to reconstruct the request and try again. A specialized SOAP fault is defined in WS-Federation to automate communicating the revised security requirements and retry of the request.

Federated Sign-Out

WS-Federation defines a federation sign-out mechanism. The purpose of federated sign-out is to indicate to federation participants that a particular federation is being terminated and they may want to clean up any cached state or security tokens for a principal that are no longer required because the principal's session is being terminated. Federated sign-out is different than token cancellation as defined in WS-Trust since federated sign-out applies to all tokens and all target sites for the principal within the federation.

Web (passive) Requestors

The WS-Federation specification defines extensions to the WS-Trust model to enable different styles of federation. The model can be used from either web service clients directly or from browsers and portals using the HTTP encoding of the WS-Trust messages. This approach enables common infrastructure and management tools irrespective of the client application platform being used.

WS-Federation defines syntax for expressing the WS-Trust protocol and WS-Federation extensions in a web browser only environment using widely supported HTTP 1.1 mechanisms (GET, POST, redirects, and cookies). WS-Federation defines encoding rules that enable many of the WS-Trust protocol extensions to be accessible via HTTP 1.1 mechanisms by standard browser clients and web applications.

The intention of this functionality is to provide a common protocol for performing Federated Identity operations for both web services and web browser based applications. A common protocol provides economies with regard to development, testing, deployment and maintenance for vendors and customers alike. Failure to integrate browser and web service scenarios will multiply these upfront costs, and will introduce even more painful and costly migration issues for customers in the future.

3. Enterprise Scenario

This scenario follows a typical large enterprise, Contoso, Ltd., that has many business relationships with external enterprises that require the protected exchange of sensitive information. Contoso provides web services for use by both their own employees and the authorized employees of their business partners. Contoso faces challenges in securing these web services, one of which is managing access by external users associated with their business partners. By utilizing WS-Federation mechanisms Contoso is able to provide access to these web services without the overhead of managing the lifecycle of user accounts for their partners' employees. This reduces administrative costs for Contoso's IT department and provides improved security. Costs are reduced because the accounts for the employees of their partners are handled by the partners themselves. As each partner employee account is managed by the user's employer, the accuracy of user's attributes asserted in claims is greatly improved because partners know the most current status of their employees. This, in turn, improves security because Contoso can make access control decisions based on the most up-to-date user context and because Contoso does not have to worry about orphaned user accounts associated with former employees of their partners that could pose security threats.

This scenario examines in detail how WS-Federation is used to dynamically configure a new federated business partner of Contoso. This scenario involves a Request for Proposal (RFP) web service operated by Contoso. The RFP web service enables Contoso to both deliver RFPs to, and receive online bids from, qualified suppliers. Contoso keeps local records about their qualified suppliers, but doesn't start from a position of configuring them up front for using their web services. In the context of the Contoso's RFP service there are two types of qualified supplier employees of interest. Any employee of the qualified supplier may query RFPs from the Contoso RFP service. Only a Bonded Employee of the qualified supplier may submit a bid against an RFP. This scenario examines how a new Contoso qualified supplier, Fabrikam, Inc., uses the RFP service for the first time. It presumes only a minimal record of the business relationship with Fabrikam represented in Contoso's database (e.g. the qualified supplier's name and an associated DNS domain name).

This scenario demonstrates the use of the Federation Metadata, Authorization Context and Indicating Specific Policy and Metadata features described in the WS-Federation specification.

When a Contract Administrator at Contoso releases a new RFP it is made available via the RFP web service. Contoso's qualified suppliers are made aware of the existence of the new RFP. How this information is made available is immaterial for the purposes of this scenario; it could be via a query web service, an RSS feed, email or the Contract Administrator could personally call their contacts at the qualified suppliers. A Fabrikam employee is one of the parties to receive notification of the new RFP. Fabrikam built a client that uses the Contoso web services. The Fabrikam client was configured independently of coordination with Contoso beyond getting the endpoint locations that provided the associated WSDL and Policy. The Fabrikam employee queries the Contoso RFP web service to retrieve the details of the RFP. The request is signed by an included token; the token is signed by Fabrikam's signing key.

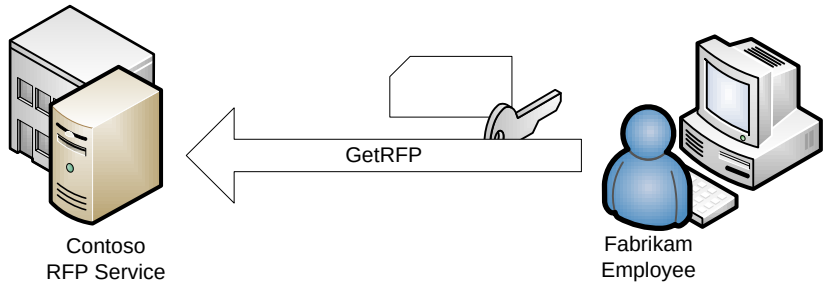

Figure 2 - RFP request

The Contoso RFP service attempts to authenticate the query. As this is the first use of the RFP service from the Fabrikam domain, additional configuration is required. The request is held while the Contoso system configures itself. Contoso's system queries its qualified supplier database to verify that Fabrikam is in fact a qualified supplier (which it is). A Federation Metadata request is issued to an endpoint in fabrikam.com. The endpoint address is constructed algorithmically from the Fabrikam domain name as described in the WS-Federation specification (see section 3.2, Acquiring the Federation Metadata Document).

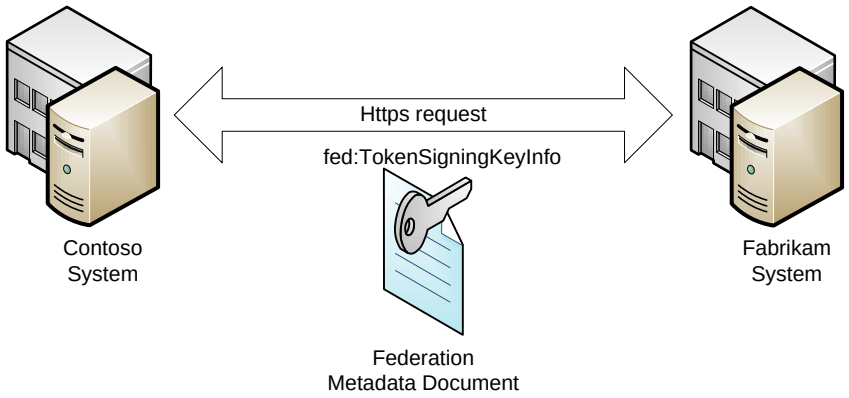

Figure 3 - Federation Metadata Document retrieval via HTTPS

```
https://www.fabrikam.com/FederationMetadata/2006-12/FederationMetadata.xml
```

Example request ereq-1

```
<fed:FederationMetadata>
  <fed:Federation>
    <fed:TokenIssuerName>
      http://www.fabrikam.com
    </fed:TokenIssuerName>
    <fed:TokenIssuerEndpoint>
      <wsa:Address>http://www.fabrikam.com/STS</wsa:Address>
    </fed:TokenIssuerEndpoint>
    <fed:TokenTypesOffered>
      ...
    </fed:TokenTypesOffered>
    <fed:SingleSignOutNotificationEndpoint>
      <wsa:Address>http://www.fabrikam.com/SignOut</wsa:Address>
    </fed:SingleSignOutNotificationEndpoint>
    <fed:UriNamedClaimTypesOffered>
```

```
      <fed:ClaimType Uri="http://schemas.xmlsoap.org/claims/EmailAddress">
        <fed:DisplayName>Email Address</fed:DisplayName>
      </fed:ClaimType>
      <fed:ClaimType Uri="http://schemas.xmlsoap.org/claims/UPN">
        <fed:DisplayName>User Principal Name</fed:DisplayName>
      </fed:ClaimType>
      <fed:ClaimType Uri="http://schemas.xmlsoap.org/claims/CommonName">
        <fed:DisplayName>Common Name</fed:DisplayName>
      </fed:ClaimType>
      <fed:ClaimType Uri="http://schemas.xmlsoap.org/claims/Group">
        <fed:DisplayName>Group</fed:DisplayName>
      </fed:ClaimType>
    </fed:UriNamedClaimTypesOffered>
    <fed:TokenSigningKeyInfo>
      <wsse:SecurityTokenReference>
        <ds:X509Data>
          <ds:X509Certificate>
            ...
          </ds:X509Certificate>
        </ds:X509Data>
      </wsse:SecurityTokenReference>
    </fed:TokenSigningKeyInfo>
  </fed:Federation>
  <!-- Signature of Federation Metadata document not shown -->
</fed:FederationMetadata>
```

Example response eresp-1

The Federation Metadata response includes a Fabrikam's signing key in a certificate under the fed:TokenSigningKeyInfo element. Contoso's system uses this key to authenticate requests from Fabrikam. Once it has the necessary information to validate that the request from Fabrikam corresponds to a qualified supplier, the Contoso system responds with the details of the requested RFP.

The federation metadata response from Fabrikam also contained the types of tokens that Fabrikam's STS offers, a federation sign out endpoint and the claim types used in tokens issued by the Fabrikam STS. How Contoso could use this additional metadata (in particular the offered claim types) will be considered later in this scenario. Claim types are not defined by WS-Federation but are normally agreed a-priori between participants in an exchange. This example uses claim types from the Federation Passive Requestor Interoperability (PRIP) document that are described in detail in Appendix A of this document.

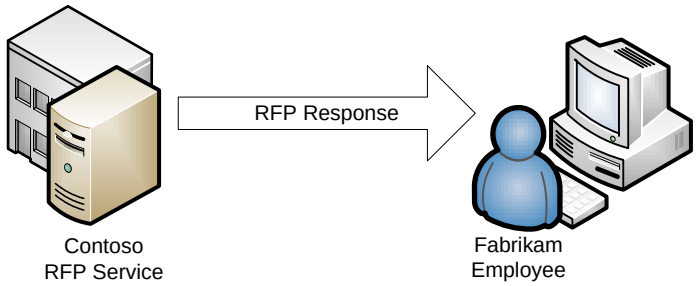

Figure 4 - RFP response

After reviewing the RFP the Fabrikam employee prepares a bid and submits it to the Contoso RFP web service. The Contoso RFP service can verify that the requestor is from fabrikam.com, but cannot verify that the requestor is a Bonded Employee of fabrikam.com. Only Bonded Employees are authorized to

submit bids. The Contoso RFP service returns a specialized SOAP fault containing specific policy and additional metadata in the fed:SpecificMetadata fault code which indicates that a claim identifying the requestor as a Bonded Employee is required.

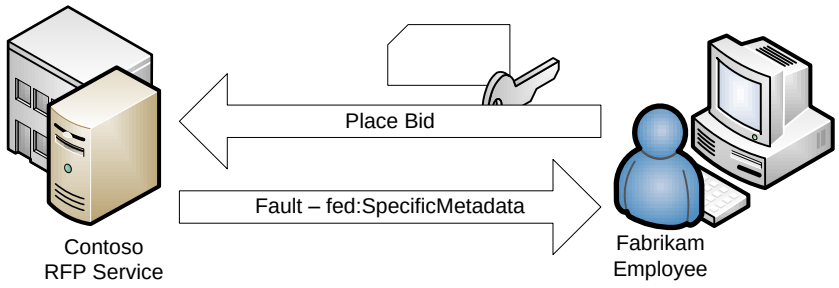

Figure 5 - Indicating Specific Policy and Metadata

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://contoso.com/RFP/PlaceBid
    </wsa:Action>
    <wsa:To>http://contoso.com/RFPService</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://client.fabrikam.com</wsa:Address>
    </wsa:ReplyTo>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body wsu:Id="body">
    <!-- application message -->
  </s:Body>
</s:Envelope>
```

Example request ereq-2

The example above is a normal SOAP application message.

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://www.w3.org/2005/08/addressing/fault
    </wsa:Action>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body>
    <s:Fault>
      <s:Code>
        <s:Value>s:Receiver</s:Value>
        <s:Subcode>
          <s:Value>fed:SpecificMetadata</s:Value>
        </s:Subcode>
      </s:Code>
      <s:Reason>
        <s:Text xml:lang="en">
          Additional credentials required in order to
          perform operation. Please resubmit request with
          appropriate credentials.
        </s:Text>
```

```
      </s:Reason>
      <s:Detail>
        <mex:Metadata>
          <mex:MetadataSection
           Dialect="http://schemas.xmlsoap.org/ws/2004/09/policy"
           Identifier="http://contoso.com/RFPService/PlaceBid">
            <!-- original token required to access RFP service-->
            <wsp:Policy>
              <sp:SymmetricBinding xmlns:sp="...">
                <wsp:Policy>
                 <sp:ProtectionToken>
                  <wsp:Policy>
                   <sp:IssuedToken>
                     ...
                   </sp:IssuedToken>
                  </wsp:Policy>
                 </sp:ProtectionToken>
                 ...
                </wsp:Policy>
               </sp:SymmetricBinding>
              <!-- additional token required to authorize bid submission-->
              <sp:EndorsingSupportingTokens>
               <wsp:Policy>
                <sp:IssuedToken>
                 <sp:Issuer>
                   <wsa:EndpointReference>
                    <wsa:Address>http://www.fabrikam.com</wsa:Address>
                  </wsa:EndpointReference>
                 </sp:Issuer>
                 <wst:Claims
                 Dialect="http://schemas.xmlsoap.org/ws/2006/12/authorization/authclaims">
                  <auth:ClaimType Uri="http://schemas.xmlsoap.org/claims/Group">
                    <auth:Value>Bonded</auth:Value>
                  </auth:ClaimType>
                 </wst:Claims>
                </sp:IssuedToken>
               </wsp:Policy>
              </sp:EndorsingSupportingTokens>
            </wsp:Policy>
          </mex:MetadataSection>
        </mex:Metadata>
      </s:Detail>
    </s:Fault>
  </s:Body>
</s:Envelope>
```

Example response eresp-2

The fault above is returned by the Contoso RFP service that indicates the specific policy and metadata required to process this request. The fault subcode of fed:SpecificMetadata informs the requestor to inspect the fault detail for the required metadata and/or policy that is described in the WS-MetadataExchange format. This example includes a WS-SecurityPolicy assertion, sp:SymmetricBinding, that contains the original policy including information such as the token required to authenticate the Requestor as an employee of a qualified supplier and thus eligible to download an RFP. This would probably include a User Principal Name (UPN) claim for audit logs. There is an additional Specific Policy

requirement, sp:EndorsingSupportingToken, for a token containing a Group claim with value “Bonded” to prove that the requestor is a Bonded Employee of Fabrikam. This claim is required to authorize submitting a bid.

The situation where additional policy and metadata is required could occur when it is difficult to advertise special conditions or due to particular application interactions. For example, if the RFP required a standard component for which Fabrikam has prepared a standard bid, the employee might pull down the RFP and submit the bid in the same session. This could cause the token provided with the bid to be missing the required Group claim to authorize submitting a bid.

The goal of using a SOAP fault to convey application-specific policy requirements is to enable automated retry of requests where possible. Clearly federation partners must agree upon the definition of claim types in the tokens they exchange. The Contoso RFP service issues the fault because it determines that the required claim can be provided by the Fabrikam STS. This can be determined since the Federation Metadata retrieved during the configuration step included the Group claim in the claim types offered. Further, the Fabrikam client can query the local STS policy to determine that it can satisfy the additional claim type required by Contoso. This enables the client to request a new token and automatically retry the operation without user intervention.

The Fabrikam client issues a request to the Fabrikam STS to obtain a token with the necessary claims. The information required to inform the STS exactly what claims are needed is provided in the auth:ClaimType element. Additionally the Fabrikam client provides additional context on why the request is being made via the auth:AdditionalContext element. This provides Fabrikam with an audit trail of not only whose tokens were issued to whom, but why. These authorization extensions to WS-Trust are described in the Authorization section of WS-Federation. These extensions to enable a rich authentication model are defined in a way that leverages the STS model of WS-Trust.

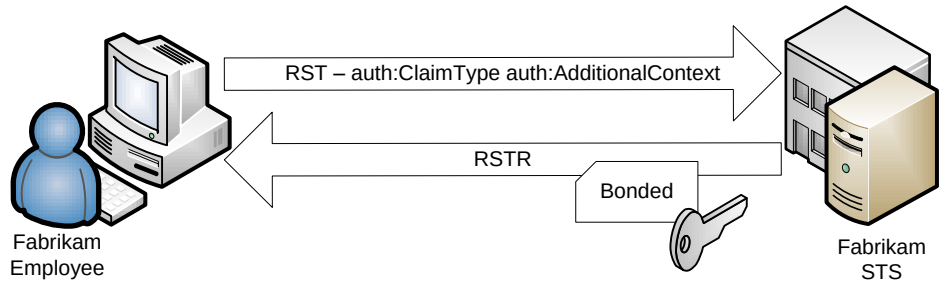

Figure 6 - Authorization Context

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
    </wsa:Action>
    <wsa:To>http://www.fabrikam.com/STS</wsa:To>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body>
    <wst:RequestSecurityToken>
      <wst:TokenType>
        ...
```

```
      </wst:TokenType>
      <wst:RequestType>
        http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
      </wst:RequestType>
     <wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2006/12/authorization/authclaims">
      <auth:ClaimType Uri="http://schemas.xmlsoap.org/claims/Group">
        <auth:Value>Bonded</auth:Value>
      </auth:ClaimType>
     </wst:Claims>
     <auth:AdditionalContext>
      <auth:ContextItem Name="http://www.fabrikam.com/rfps/tracking/for">
        <auth:Value>Contoso</auth:Value>
      </auth:ContextItem>
      <auth:ContextItem Name="http://www.fabrikam.com/rfps/tracking/bidid">
        <auth:Value>t9000hx-1139</auth:Value>
      </auth:ContextItem>
      <auth:ContextItem Name="http://www.fabrikam.com/rfps/tracking/value">
        <auth:Value>$1,320,000.00</auth:Value>
      </auth:ContextItem>
     </auth:AdditionalContext>
    </wst:RequestSecurityToken>
  </s:Body>
</s:Envelope>
```

Example request ereq-3

In the above RST message example the missing claim type being requested is included in the wst:Claims element defined by WS-Trust. Note that the dialect being used is the Common Claims Dialect from WS-Federation. The specific claim type is an example based on the PRIP Group claim type described in Appendix A. This example also illustrates the use of the Authorization extension to WS-Trust that is defined by WS-Federation in the auth:AdditionalContext element. Here simple context item and values are illustrated; the semantics of these are not defined by WS-Federation.

In this example the additional authorization context is not shared with Contoso; this is completely within Fabrikam's realm. This provides a very useful extension for making decisions about issuing tokens and auditing that issuance. For example, in this case Fabrikam could trigger additional processing before issuing the Bonded Group claim if the value of the RFP for Contoso was above a threshold approved for the employee making the request. Similarly Fabrikam could log requests for bonded claims that are issued for its employees including which of their business partners the token was to be used for.

In this case the request is successful and the Fabrikam STS mints a token with the necessary claim and returns it to the client. The Fabirkam client now resubmits the bid with both the original token and the new token to the Contoso RFP Web service. Optionally, the STS could have issued one new token with all the required claims.

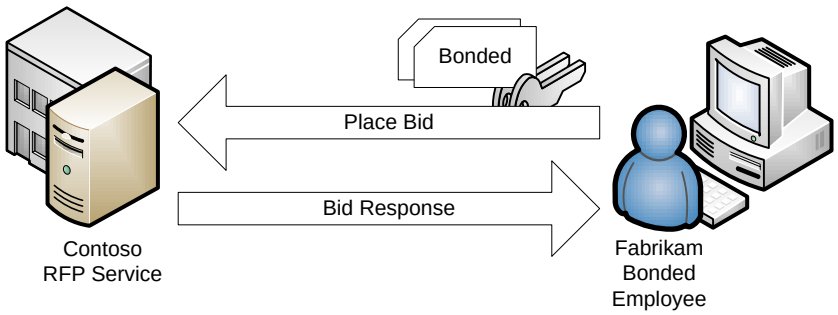

Figure 7 - Request from Figure 5 remade with proper claims

3.1 Enterprise Scenario Summary

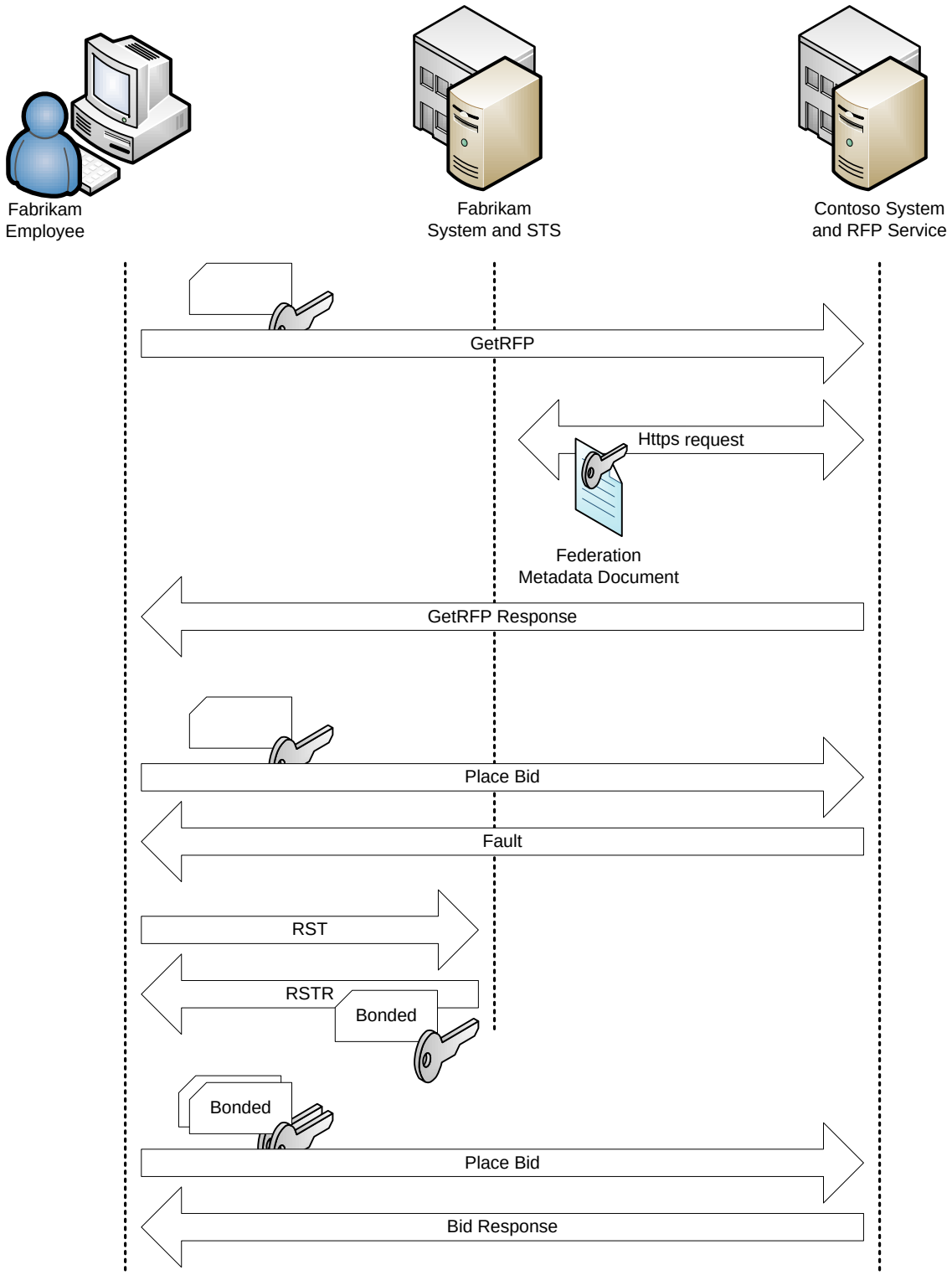

Figure 8 - Enterprise scenario overview

In review, this scenario begins when two enterprises decide to enter into a federation. This is a business level decision, not a technological one. This federation is enabled by the use of WS-Federation and it's extensions to the WS-Trust Security Token Model. Starting from the top of Figure 8, the initial request is made from a Fabrikam Requestor to the Contoso RFP service, a Resource Provider. This request is secured with a security token issued by the Fabrikam STS. As this is the first request from the Fabrikam domain to a Contoso Resource Provider this triggers a request, to an algorithmically determined endpoint at the Fabrikam system, to retrieve its Federation Metadata document. Contoso uses this to configure its RFP service to accept security tokens issued from the Fabrikam STS. The Contoso Resource Provider can now verify the security credentials of the Fabrikam Requestor and returns the GetRFP response.

Next a Fabrikam employee places a bid to the Contoso Resource Provider that is secured with a token from the Fabrikam STS. In the example, the original token is missing an additional Group claim that identifies the Requestor as a Bonded Employee, which Contoso requires to authorize submitting a bid. The Contoso Resource Provider returns a fault which indicates a specific policy that must be met, conveying the missing claim. The Fabrikam requestor processes this information and makes a request for a token with the identified claim to the Fabrikam STS. This request uses the authorization context extensions defined by WS-Federation to provide additional information to process the request for the token. The request is successful and a token is returned to the Contoso Requestor that contains the required claim.

The Fabrikam requestor now places the bid to the Contoso Resource Provider secured with the original token and carrying the new token with the required claim. The Contoso Resource Provider is now satisfied and issues a bid response.

Contoso has another service that it provides to its partners, a web page to check bid status. This web page requires that any access must be authenticated so that Contoso's partners only see the status of their own RFP bids. This is done by requiring a token from the requestor's Identity Provider (IP)/ Security Token Service (STS) that the Contoso web site can authenticate and then limit access based on the user. This exchange is enabled through the use of the WS-Federation Web (passive) Requestor protocol (see section 13 of [WS-Federation]). Essentially this is a web front end to the same infrastructure in which WS-Trust messages are constructed from web pages and transmitted via the standard HTTP protocol. This exchange has a variety of styles to provide rich support for complex requirements requiring significant interactions to compact forms for low bandwidth situations. What follows is a relatively simple example.

A Fabrikam employee who is signed in within Fabrikam's network launches their browser to check on their bid on the Contoso RFP bid status page. The Contoso system requires a token from the Fabrikam STS and returns a HTTP redirect message to the browser. This causes the Fabrikam employee's browser to send an HTTP GET message with the WS-Trust request to the Fabrikam STS for the required token. At this point the Fabrikam STS could present a GUI to authenticate, but in this example the STS has cached information and is able to authenticate the request without further user intervention. The Fabrikam STS returns the requested token to the Fabrikam employee's Relying Party, the Contoso system, via an HTTP

POST which contains the WS-Trust response. The Contoso system is able to authenticate the request and returns the RFB bid statuses for Fabrikam.

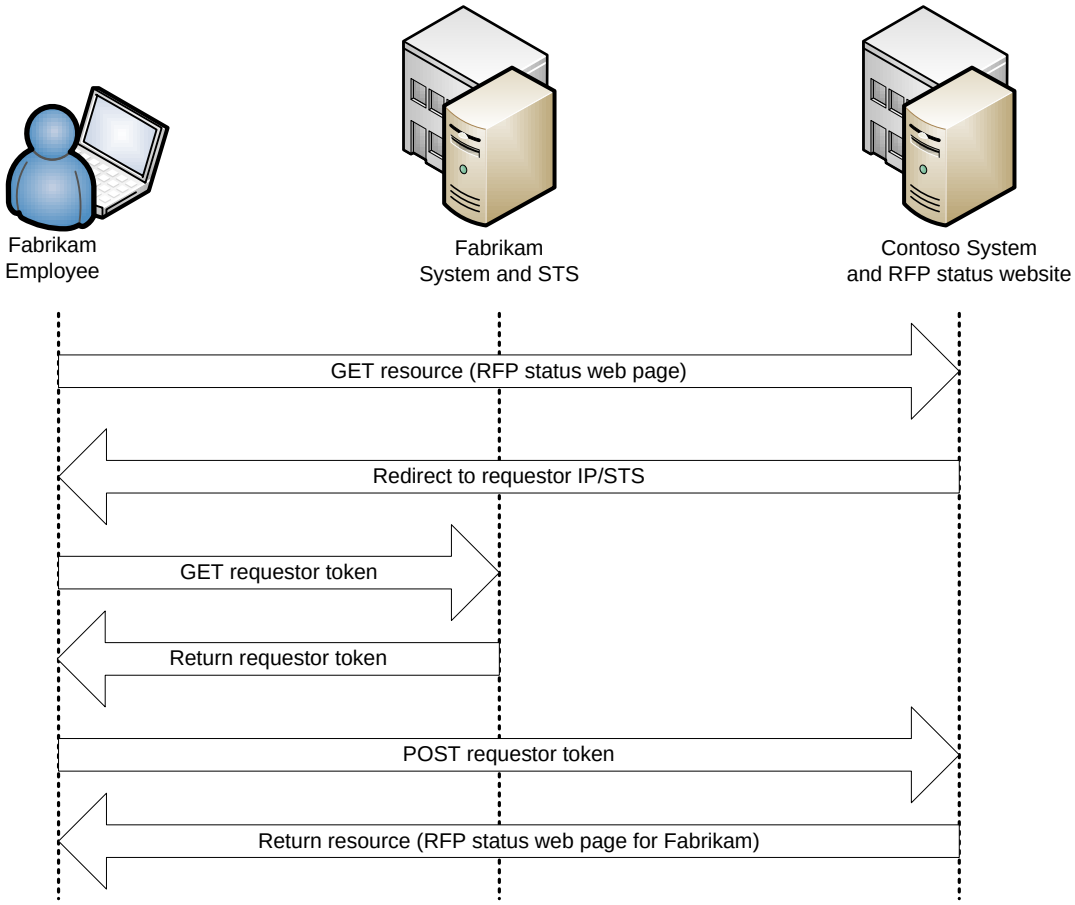

Figure 9 - Web (passive) requestor

The above diagram depicts the flow of requests between the Fabrikam employee's browser, the Contoso system and the Fabrikam STS. This flow is composed of standard HTTP requests, redirects and posts so no special browser or plugin is required for the Fabrikam employee to use. There are a number of HTTP parameters at play in this exchange that are beyond the scope of this paper to cover. These HTTP parameters facilitate mapping WS-Trust messages into plain HTTP requests, redirects and responses. It is the combination of the standard HTTP mechanisms and the HTTP parameters defined by WS-Federation that enable the extension of the WS-Trust and the WS-Federation extensions for normal web browser clients. Section 13 of WS-Federation covers this functionality in detail.

4. Healthcare Scenario

In this scenario there is a Medical Authority that can legally certify doctors, hospitals and other public healthcare facilities. This Medical Authority is authorized to play such role by a regional or national body. It might also have recognition internationally through formal relationships with other national and international medical associations. This Medical Authority defines widely recognized practices, documents and claims within its area of jurisdiction. There are real benefits to healthcare providers and practitioners to participating within the frameworks defined by the Medical Authority. There are naturally many, many participants within this ecosystem. It is not feasible for all of them to know of each other in advance. There are many instances where it is useful to be able to dynamically recognize other valid participants and get (provide) access to resources from (to) them. Given the nature of the information in this realm, access to critical care information can equate to real life and death decisions.

Based on feedback from its members the Medical Authority has decided to provide services to its members that allow them to access information in a more timely manner and enable new collaboration scenarios that are unlocked by the use of web services based technology. One request from the members is that the Medical Authority utilizes its recognized role as a legally certifying body by acting as a root Certificate Authority (CA) and providing a Security Token Service (STS) to issue its members' credentials to use in their practices. All participants within this environment will accept certificates issued from this CA and its delegates. As a CA the Medical Authority issues tokens to physicians that have passed its board certification and handles revocation of those tokens should a physician's status change.

A second request from members is that the Medical Authority establish relationships with other medical facilities, such as hospitals and clinics, to allow them to work together to provide more integrated information services to improve patient care. The Medical Authority accomplishes this by enabling medical facilities to act as delegate CAs of the Medical Authority. This allows the medical facilities to issue tokens to verified health care practitioners within the scope of each medical facility's local operations but within the overall trust fabric provided by the Medical Authority. This enables a trusted network to be established within the ecosystem allowing each participant to make more informed local trust decisions.

The Medical Authority also defines a federated Health Record Service framework for which its members can implement conformant services and clients. This Health Record Service framework defines document formats and common claims for integrating health information services within the Medical Authority's jurisdiction. The Medical Authority itself does not store or aggregate any of these records, that is the responsibility of the individual participants in the ecosystem. This framework empowers the individual participants in the system to make decisions about how and when to release Health Records based on a common set of agreed upon policies.

This scenario examines how different parties can, through WS-Federation mechanisms, express their requirements for using the Health Record Service and make decisions about access across different trust realms without a priori setup.

With that context, this scenario examines the use of WS-Federation as the mechanism by which these services can be established including the sharing of Federation Metadata, protecting Privacy and Indicating Specific Policy and Metadata. It also illustrates further use of the Common Claims Dialect.

In this scenario when an ER Doctor arrives at work he signs into the hospital's system (regardless of platform, form factor, or vendor) that provides him access to all the services he will need. At this point, a request is made to the Medical Authority STS. This request may include the authentication material the ER Doctor presented to his sign on prompt, and it requests that a token T_1 with key K_1 be issued from the root Medical Authority CA. This token, T_1, contains associated claims that the ER Doctor is a licensed physician by the Medical Authority. The ER Doctor is also issued a token T_2 by the Hospital's STS. Token T_2 has associated K_2 from the Hospital's CA as a result of a successful login and contains claims that can be vetted by the local hospital, i.e. indicating that the ER Doctor is on duty. So far these are normal WS-Trust interactions. The ER Doctor does not need to be actively engaged in acquiring these credentials, since the entry system manages these as part of its operations.

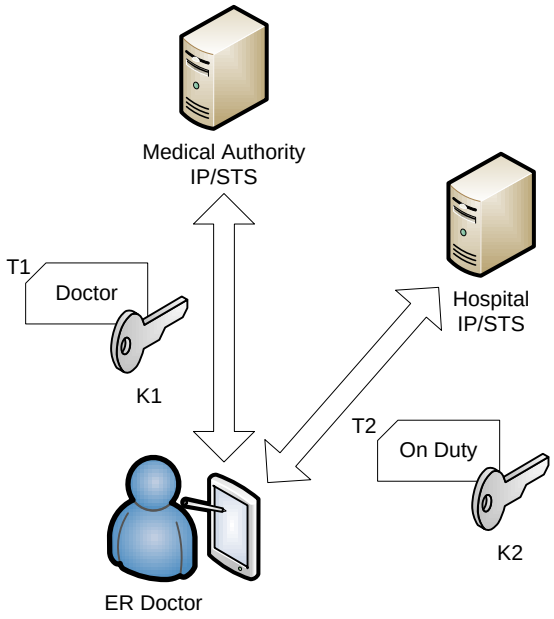

Figure 10 - Multiple Identity Providers

During the ER Doctor's shift an unconscious patient, Carol, is admitted into the Hospital. The ER Doctor needs to ascertain what if any conditions Carol has in order to properly administer the correct treatment from the available options. Carol has a life threatening allergy to one of the options the ER Doctor is considering. This information is not available to the ER doctor though it is recorded in her health records.

The Hospital staff found a student ID card on Carol with a link to the affiliated University Hospital's Health Record Service that participates in the framework defined by the Medical Authority. Before administering any medication the University Hospital's Health Record Service link is entered into the Hospital's system. The Hospital's system initiates a WS-MetadataExchange request to the University Hospital's Health Record Service URL. This returns two available endpoints for requesting medical records, one for normal access and the other for emergency access. The response also contains the associated access policies for each endpoint in the returned Federation Metadata Document. The ER Doctor does not need to be aware of these protocol exchanges; all he sees are 2 options for searching for Carol's information

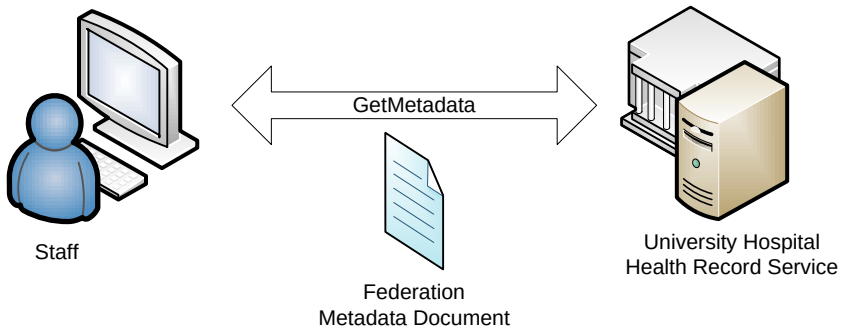

Figure 11 - Federation Metadata Document retrieval via WS-MetadataExchange

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Request
    </wsa:Action>
    <wsa:To>http://healthservices.university.edu</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://client.hospital.org</wsa:Address>
    </wsa:ReplyTo>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body />
</s:Envelope>
```

Example request hreq-1

Above we see a generic WS-MetadataExchange GetMetadata request to the University's Health Record Service.

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Response
    </wsa:Action>
    <wsa:To>http://client.hospital.org</wsa:To>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body>
    <mex:Metadata>
      <!-- WSDL, Schema and Policy sections not shown for brevity -->
      <!-- Each service defined here will have its own policy
           for required calims, tokens etc. Not anyone with a token
           from the issuer below can use all services.
      -->
      <mex:MetadataSection
        Dialect="http://schemas.xmlsoap.org/ws/2006/12/federation">
        <fed:FederationMetadata>
          <fed:Federation>
            <fed:TokenIssuerName>
              http://medicalauthority.org
            </fed:TokenIssuerName>
            <!-- Required token & claim types are specified in Policy -->
```

```
            <fed:TokenIssuerEndpoint>
              <wsa:Address>
               http://medicalauthority.org/federation/STS
              </wsa:Address>
            </fed:TokenIssuerEndpoint>
            <fed:SingleSignOutNotificationEndpoint>
              <wsa:Address>
               http://healthservices.university.edu/Federation/SignOut
              </wsa:Address>
            </fed:SingleSignOutNotificationEndpoint>
          </fed:Federation>
        <!-- Signature of Federation Metadata document not shown -->
        </fed:FederationMetadata>
      </mex:MetadataSection>
    </mex:Metadata>
  </s:Body>
</s:Envelope>
```

Example response hresp-1

The response to the GetMetatadata request is detailed above. This response contains WSDL, Schema and Policy information in addition to a WS-Federation Metadata Document. Note that each service defined here will have its own policy for required claims, tokens etc. A token from the named token issuer is required to use any of the services.

The ER Doctor selects the emergency access path. The policy for use of the University Hospital's emergency Health Record Service endpoint requires that a token of a physician registered with the Medical Authority be presented which is indicated by fed:TokenIssuerName in the Federation Metadata document returned in the previous request shown above. Unfortunately the request receives a fault. This fault specifies via the fed:SepcificMetadata fault code that a third token T_3 is required. This is a token required from someone authorized to release Carol's records. The specific metadata returned in the fault contains a reference to the Endpoint Reference (EPR) of Carol's Primary Care Provider who is authorized to release these records. This reference is contained in the returned metadata section within the fault detail, specifically the fed:TokenIssuerEndpoint.

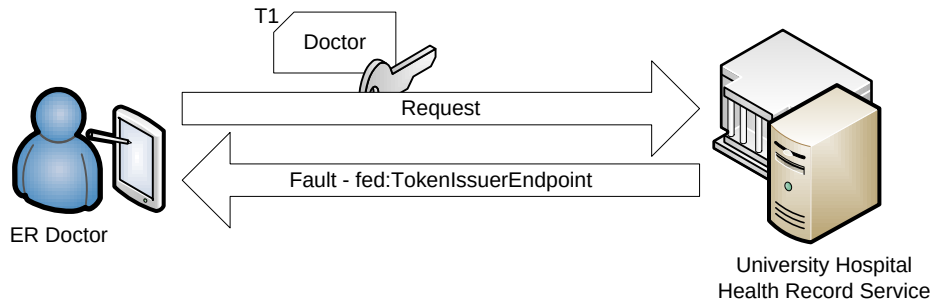

Figure 12 - Indicating Specific Policy and Metadata

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
```

```
      http://medicalauthority.org/EmergencyAccess
    </wsa:Action>
    <wsa:To>http://healthservices.university.edu/EmergencyAccess</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://client.hospital.org</wsa:Address>
    </wsa:ReplyTo>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body wsu:Id="body">
    <!-- application message -->
  </s:Body>
</s:Envelope>
```

Example request hreq-2

Above we see the application request to the University Health Record service.

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://www.w3.org/2005/08/addressing/fault
    </wsa:Action>
  </s:Header>
  <s:Body>
    <s:Fault>
      <s:Code>
        <s:Value>s:Receiver</s:Value>
        <s:Subcode>
          <s:Value>fed:SpecificMetadata</s:Value>
        </s:Subcode>
      </s:Code>
      <s:Reason>
        <s:Text xml:lang="en">
          Additional credentials required in order to
          perform operation. Please resubmit request with
          appropriate credentials.
        </s:Text>
      </s:Reason>
      <s:Detail>
        <mex:Metadata>
          <mex:MetadataSection
           Dialect="http://schemas.xmlsoap.org/ws/2004/09/policy"
           Identifier="http://healthservices.university.edu/EmergencyAccess">
           <wsp:Policy>
            <!-- Original Policy not shown for brevity -->
            <!-- Additional Policy required-->
             <sp:EndorsingSupportingTokens>
              <wsp:Policy>
              <sp:IssuedToken>
               <sp:Issuer>
                <wsa:EndpointReference>
                 <wsa:Address>http://PrimaryCareProvider.com</wsa:Address>
                </wsa:EndpointReference>
               </sp:Issuer>
               <wst:Claims
               Dialect="http://schemas.xmlsoap.org/ws/2006/12/authorization/authclaims">
                 <auth:ClaimType
                   Uri="http://claims.medicalauthority.org/MedicalHistory"/>
```

```
                </wst:Claims>
                <sp:RequestSecurityTokenTemplate>
                  <auth:AdditionalContext>
                   <auth:ContextItem
                     Name="healthservices.university.edu/AccessReq">
                     <auth:Value>http://client.hospital.org</auth:Value>
                   </auth:ContextItem>
                   <auth:ContextItem
                     Name="healthservices.university.edu/Patient">
                     <auth:Value>Carol</auth:Value>
                   </auth:ContextItem>
                  </auth:AdditionalContext>
                </sp:RequestSecurityTokenTemplate>
               </sp:IssuedToken>
               </wsp:Policy>
              </sp:EndorsingSupportingTokens>
             </wsp:Policy>
            </mex:MetadataSection>
          </mex:Metadata>
        </s:Detail>
      </s:Fault>
    </s:Body>
</s:Envelope>
```

Example response hresp-2

The fault that the University Hospital Health Record service returns is detailed above. It indicates that there is an additional token required for this request to access Carol's records. The fault subcode of fed:SpecificMetadata informs the requestor to inspect the fault detail for the required metadata and/or policy that is described in the WS-MetadataExchange format. The WS-SecurityPolicy assertion in the example, sp:EndorsingSupportingToken, shows the required issuer and claim values. This example also shows use of the WS-SecurityPolicy RequestSecurityTokenTemplate assertion. This requires the Hospital system to pass the Authorization auth:AdditionalContext structure contained within the assertion to the indicated token issuer in the SecondaryParameters element on an RST as defined by WS-Trust. The Hospital is not required to understand these claims. This allows additional context about the request for the token to be conveyed from the Resource Provider, in this instance the University Hospital, to the Primary Care Provider's STS. This example includes information about which hospital is making a request for health records and for which patient. The Primary Care Provider's STS can use this information to assist it in making a decision about authorizing the request or for auditing.

How did the University Hospital know that additional information was required for emergency access to Carol's records? When Carol enrolled in the University she chose to enroll in the University's health insurance program administered by the University with services provided by the University Hospital. Upon enrolling in this program she was asked if she would allow physicians licensed by the Medical Authority to be granted access to her health record information in emergency situations. Being careful and having more trust in her home town Primary Care Provider (PCP) she said no to the general emergency release request, but identified her PCP as the point of contact in an emergency. If she had said yes, the above request would have worked. She said no with the knowledge that her PCP could provide for emergency release of her records if needed. She authorized her PCP as a delegate on her behalf should an emergency request for her records come in. She provided contact information which included the link to her PCP's Health Record Service. Federation metadata was then used to exchange information between the University Hospital and the PCP. This allows the University to accept tokens

issued by the PCP's service for emergency access and to indicate to requestors when such a token is required.

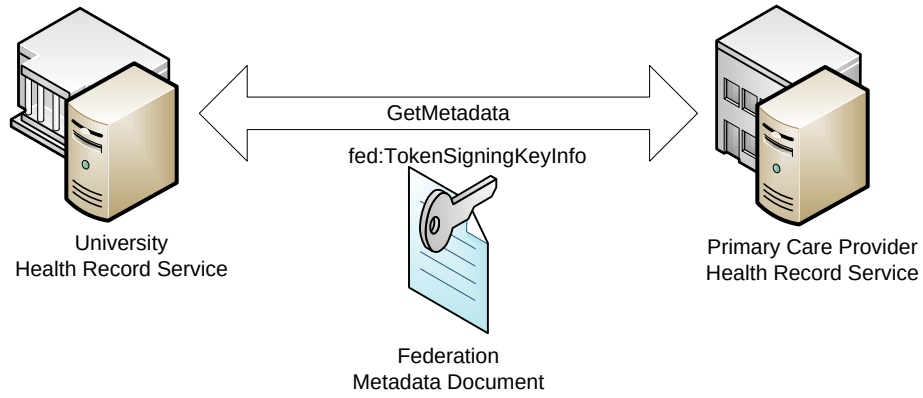

Figure 13 - Federation Metadata Document retrieval via WS-MetadataExchange

Returning to where the scenario left off at the Hospital, the ER Doctor has requested information about Carol. Upon receiving the fault that indicates the need for authorization from Carol's Primary Care Provider, the application automatically initiates a new WS-Transfer Get request to the Federation Metadata path. This request is constructed using the Issuer End Point Reference (EPR) information returned in the SOAP fault. This service might be hosted as, generally speaking, physicians' offices are not yet typically set up to provide the level of access required in these types of exchanges. The Hospital's application is capable of evaluating the returned interface and policies of the Provider's service. The Hospital's application recognizes that it can request the Primary Care Provider's STS to issue a new credential T_3 for the release of patient records.

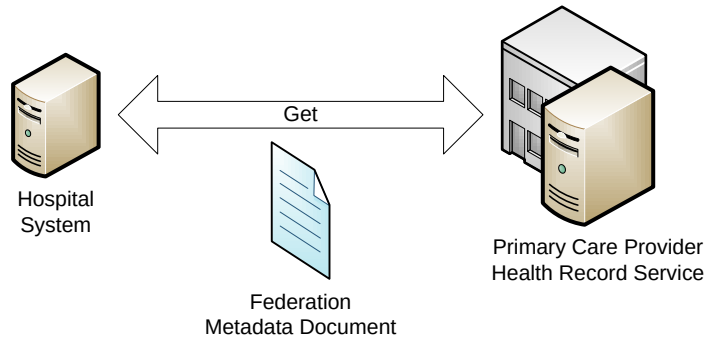

Figure 14 - Federation Metadata Document retrieval via WS-Transfer

```
<s:Envelope xmlns:s="..." xmlns:wsa="...">
  <s:Header>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
    </wsa:Action>
    <wsa:To>http://PrimaryCareProvider.com/mex</wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>http://client.hospital.org</wsa:Address>
    </wsa:ReplyTo>
    <!-- Other headers not shown for brevity -->
```

```
  </s:Header>
  <s:Body />
</s:Envelope>
```

Example request hreq-3

The example above illustrates a generic WS-Transfer Get request to the metadata endpoint of the Primary Care Provider.

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
    </wsa:Action>
    <wsa:To>http://client.hospital.org</wsa:To>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body>
    <mex:Metadata>
      <mex:MetadataSection Dialect="http://schemas.xmlsoap.org/wsdl/">
        <!-- Not shown for brevity -->
      </mex:MetadataSection>
      <mex:MetadataSection
        Dialect="http://schemas.xmlsoap.org/ws/2004/09/policy"
        Identifier="http://PrimaryCareProvider.com/STS/policy">
        <!-- STS Token Issuance EPR Policy -->
        <wsp:Policy>
         <sp:SymmetricBinding>
           <wsp:Policy>
            <sp:ProtectionToken>
             <wsp:Policy>
             <sp:IssuedToken>
               <sp:IssuerName>http://MedicalAuthority.org</sp:IssuerName>
               <wst:Claims
               Dialect="http://schemas.xmlsoap.org/ws/2006/12/authorization/authclaims">
                <auth:ClaimType
                  Uri="http://claims.medicalauthority.org/Doctor"/>
                <auth:ClaimType
                  Uri="http://claims.medicalauthority.org/OnDuty"/>
               </wst:Claims>
             </sp:IssuedToken>
             </wsp:Policy>
            </sp:ProtectionToken>
            ...
           </wsp:Policy>
         </sp:SymmetricBinding>
        </wsp:Policy>
      </mex:MetadataSection>
      <mex:MetadataSection
       Dialect="http://schemas.xmlsoap.org/ws/2006/12/federation">
        <fed:FederationMetadata>
          <fed:Federation>
            <fed:TokenIssuerName>
              http://primaryCareProvider.com
            </fed:TokenIssuerName>
            <fed:TokenIssuerEndpoint>
              <wsa:Address>
```

```
                http://primaryCareProvider.com/STS
              </wsa:Address>
            </fed:TokenIssuerEndpoint>
            <fed:TokenTypesOffered>
              <fed:TokenType Uri="..."/>
            </fed:TokenTypesOffered>
            <fed:SingleSignOutNotificationEndpoint>
              <wsa:Address>
                http://primaryCareProvider.com/SignOut
              </wsa:Address>
            </fed:SingleSignOutNotificationEndpoint>
            <fed:UriNamedClaimTypesOffered>
              <fed:ClaimType
                Uri="http://claims.medicalauthority.org/PsychiatricHistory">
                <fed:DisplayName>
                 Psychiatric History Record Locator
                </fed:DisplayName>
              </fed:ClaimType>
              <fed:ClaimType
                Uri="http://claims.medicalauthority.org/MedicalHistory">
                <fed:DisplayName>
                 Medical History Record Locator
                </fed:DisplayName>
              </fed:ClaimType>
            </fed:UriNamedClaimTypesOffered>
          </fed:Federation>
          <!-- Signature of Federation Metadata document not shown -->
        </fed:FederationMetadata>
      </mex:MetadataSection>>
    </mex:Metadata>
  </s:Body>
</s:Envelope>
```

Example response hresp-3

The returned metadata contains the WSDL, policy and Federation Metadata of the Primary Care Provider's services including its STS. The federation metadata document also contains claims that it provides, seen above in the UriNamedClaimTypesOffered element. Two of the claim types identified here, MedicalHistory and PsychiatricHistory, are claim types that the Medical Authority has defined. The Hospital and University Hospital Health Record Service, being participants in the Medical Authority's ecosystem, recognize them and understand their significance with respect to privacy of patient records.

The Primary Care Provider STS requires two tokens for authorizing a request to issue T_3. One is a token issued by the Medical Authority of a licensed physician as indicated by a contained claim; this can be satisfied by T_1. The other is a token containing a claim of OnDuty of a medical facility that is certified by the same Medical Authority; this can be satisfied by T_2. This second token is required to prevent access by physicians who are not acting on official business. The only direct trust relationship the PCP has is with the Medical Authority. In this example, the PCP has never interacted with the Hospital before.

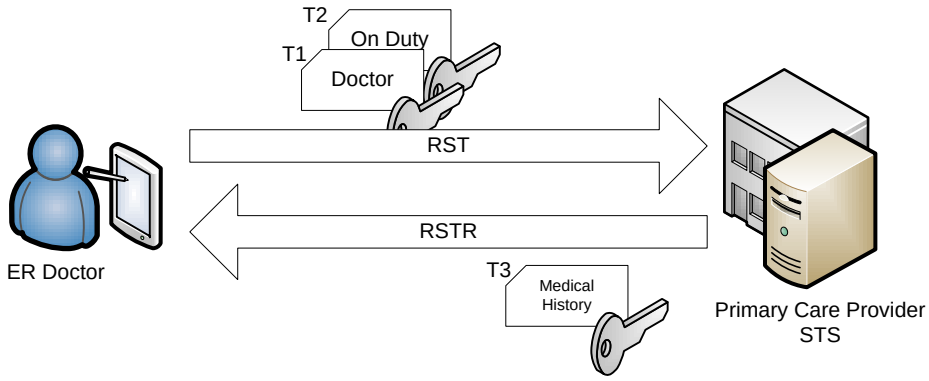

Figure 15 - Requesting a Security Token

The Hospital has its own requirements in this scenario. The Hospital does not want any exposure to the details of T_3 that may be sensitive. In this instance it is familiar with an offered claim type from the Primary Care Provider, Psychiatric History, which could be very sensitive and is not relevant to the Hospital's needs. If this claim is issued, the Hospital wants to be protected so that only the PCP and University (presumably) can access it. To provide a valid log of information, the hospital does have a strong requirement that it have a clear record of who used its services to request records and that the correct patient information was requested. The Hospital uses the confidential tokens feature from WS-Federation to accomplish this. In the RST request to the Primary Care Provider's STS the sensitive claims related to Carol's information are specified by the Hospital in the priv:ProtectData element. This requires confidentiality be applied to the sensitive claim, if issued, in the response so that only the PCP and University Hospital (presumably) can access it.

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
    </wsa:Action>
    <wsa:To>http://primaryCareProvider.com/STS</wsa:To>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body wsu:Id="rst">
    <wst:RequestSecurityToken>
     <wst:TokenType>
        ...
     </wst:TokenType>
     <wst:RequestType>
        http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
     </wst:RequestType>
     <wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2006/12/authorization/authclaims">
        <auth:ClaimType Uri="http://claims.medicalauthority.org/MedicalHistory"/>
     </wst:Claims>
     <wst:SecondaryParameters>
      <auth:AdditionalContext>
        <auth:ContextItem Name="healthservices.university.edu/AccessReq">
          <auth:Value>http://client.hospital.org</auth:Value>
        </auth:ContextItem>
        <auth:ContextItem Name="healthservices.university.edu/Patient">
          <auth:Value>Carol</auth:Value>
        </auth:ContextItem>
      </auth:AdditionalContext>
```

```
      </wst:SecondaryParameters>
      <priv:ProtectData>
       <wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2006/12/authorization/authclaims">
         <auth:ClaimType Uri="http://claims.medicalauthority.org/PsychiatricHistory"/>
       </wst:Claims>
      </priv:ProtectData>
     </wst:RequestSecurityToken>
  </s:Body>
</s:Envelope>
```

Example request hreq-4

When T_3 is issued in response to this request the claims within priv:Protected data can only be accessed by the PCP and the University Hospital. The Hospital will not be able to access the information even though the token is being returned to them. The example request above includes the use of the Secondary Parameters carrying the Authorization auth:AdditionalContext as required by hresp-2 above.

Upon receipt of T_3 the Hospital application resubmits the original request to the emergency University Hospital Health Record Service endpoint, this time with T_1 and T_3. The request is successful and Carol's health records are returned to the Hospital. The ER Doctor quickly identifies Carol's allergy to one of the medications he was considering using and chooses the safer alternative.

4.1 Healthcare Scenario Summary

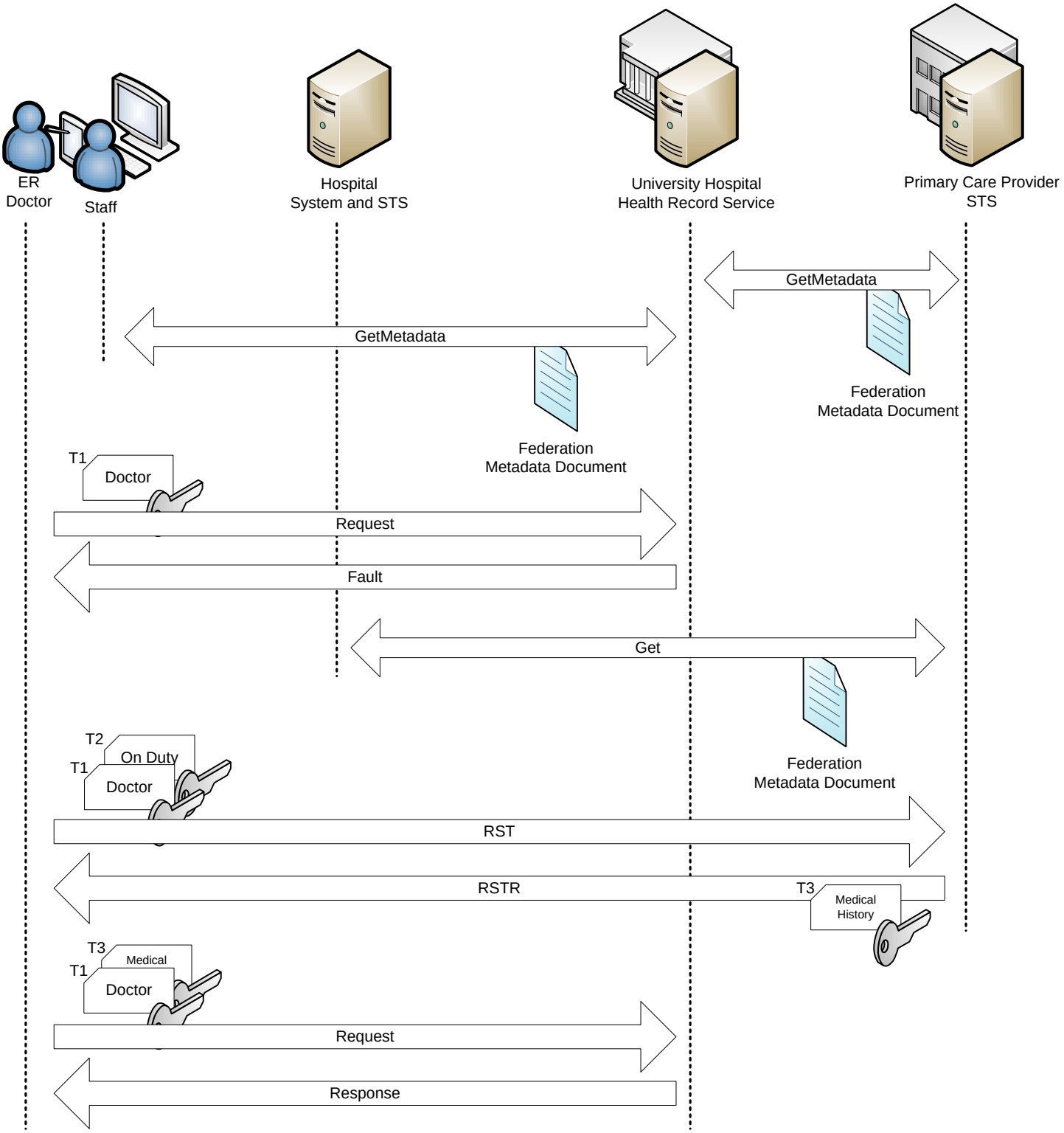

Figure 16 - Healthcare scenario overview

This scenario begins when an ER Doctor arrived and signed in at the Hospital where he is employed. At that time he is issued two tokens, one that indicates he is a licensed physician and another that indicates he is on duty. These tokens are issued with credentials within the Medical Authority's certificate chain. This is depicted in Figure 10 at the beginning of this scenario. Credentials issued within that certificate chain are widely trusted by participants that use services defined by, but not managed by, the Medical Authority including a Health Record Service. While the ER Doctor is on duty an unconscious patient is admitted to the ER. She has an allergy to one of the medications that could be used to treat her that is recorded in her health records, but this fact is not known by the ER Doctor.

This is where the scenario starts in Figure 16 above. When Carol, the patient, enrolled in the University's health insurance program she was asked if she wanted to allow emergency access to her health records from the University Hospital by licensed Medical Authority physicians. She was not comfortable with this and said no. She did authorize her Primary Care Provider (PCP) as a delegate for releasing those records and provided the University with contact information. The University Hospital and the PCP entered into a federation and enabled the sharing of tokens by trading WS-Federation Metadata Documents using WS-MetadataExchange. This is shown at the upper right of Figure 16.

The Hospital Staff finds a student identification card Carol is carrying which includes a link printed on it to her University Hospital's Health Record Service location. The Hospital Staff enter this link into their system and a WS-MetadataExchange request is made to the University Hospital system. The returned metadata provides alternate endpoints to access health records; given the circumstances, the Hospital Staff choose the emergency access option. This requires a token of a physician currently licensed by the Medical Authority. The ER Doctor logs in so that his token T_1 can be used. A specialized fault defined by WS-Federation is returned that indicates specific metadata is required to process this claim, namely a token issued by the PCP.

The Hospital System automates a WS-Transfer request to the PCP endpoint to retrieve the required policies for requesting the needed token. In the returned metadata there is an offered claim type that is defined by the Medical Authority that is very sensitive and not needed by the hospital to administer treatment. The Primary Care Provider STS requires two tokens to issue the required one, one token of a physician currently licensed by the Medical Authority and another indicating the physician is on duty. This is to prevent unnecessary requests from being processed as emergencies and to protect the privacy of patient information. These are more stringent requirements than the University Hospital has. The ER Doctor logs in and releases the required tokens (issued when he arrived at work as shown in Figure 10). The RST request is accepted and the required token T_3 is returned to the Requestor.

Now the request to the University Hospital Health Record Service is made again, this time with T_1 and T_3. The request is accepted and Carol's health records are returned.

The ER Doctor quickly identifies Carol's allergic condition and chooses a safe alternative treatment.

The next portion of this scenario examines what happens when the ER Doctor signs out at the Hospital. The federation metadata documents above include a fed:SingleSignOutNotificationEndpoint. This is the endpoint to which fed:SignOut messages can be sent to participants in a federation to indicate that any state and tokens related to the principal are no longer required.

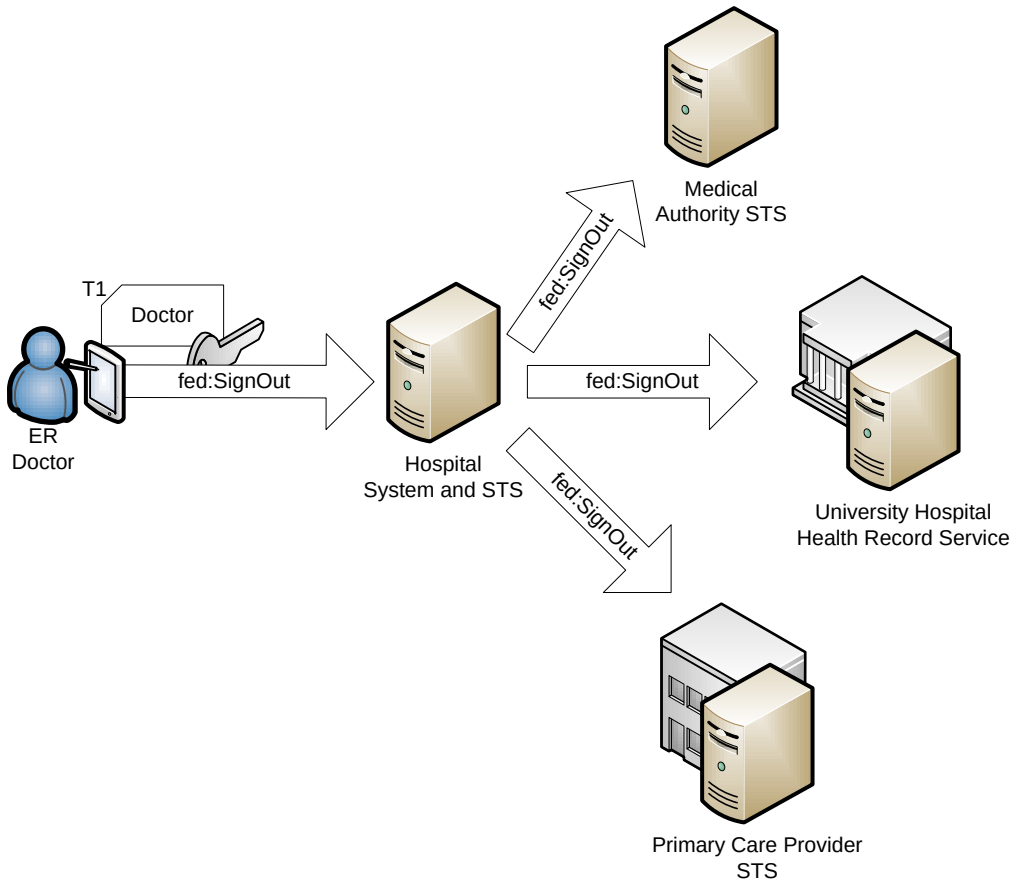

Figure 17 - Sign out

As illustrated above, the Hospital STS is where the initial fed:SignOut message is sent when the ER Doctor signs out. That system tracked the various systems signed into by the ER Doctor throughout the day. When the Hospital STS receives the fed:SignOut message it sends fed:SignOut messages to all of the sign out endpoints of relevant token and target services it knows about. The fed:SignOut message is secured by the ER Doctor's token so that others cannot maliciously send this request to close his active sessions. The token is not shown for the federated sign out message in the figure above or in the example below. The token (or tokens) is in or referred from within the fed:SignOutBasis element. There is no restriction on the token types allowed.

```
<s:Envelope>
  <s:Header>
    <wsa:Action>
      http://schemas.xmlsoap.org/2006/12/Federation/SignOut
    </wsa:Action>
    <wsa:To>http://sts.hospital.org</wsa:To>
    <!-- Other headers not shown for brevity -->
  </s:Header>
  <s:Body>
    <fed:SignOut>
      <fed:SignOutBasis>
        <!-- Token or token reference of principal signing out -->
      </fed:SignOutBasis>
```

```
    </fed:SignOut>
  </s:Body>
</s:Envelope>
```

Example request hreq-5

Appendix A – PRIP Claim Types

The WS-Federation Passive Requestor Interoperability Profile was created and distributed to participants involved in WS-Federation interoperability testing. Within that document there were common claim types defined. This appendix contains those claim types and their associated mappings to WS-Federation's Offered Claim Types and Common Claims Dialect as they are used in examples in this document. To illustrate mapping of claims to a token, the mapping to SAML 1.1 assertions, as provided in the original PRIP document, is also provided. Normative language from the PRIP document is carried forward here; this document remains non-normative.

The claim types supported as part of the Passive Requestor Interoperability Profile are defined below. They all follow the name/value pattern. However, four claim types have been identified through usage that provides specific and useful semantics.

- EmailAddress
- UPN
- CommonName
- Group

One extensibility claim type is also supported.

- NameValue

Email Address Claim Type

EmailAddress claim type is used to identify a specific security principal via an email address, and conforms to "addr-spec" as defined in IETF RFC 2822. The value MUST be unique such that it can be used for identification and authorization.

Offered Claim Type Example

```
<!-- Example shows use of a EmailAddress claim in Offered Claim type -->
<fed:UriNamedClaimTypesOffered>
  <fed:ClaimType Uri="http://schemas.xmlsoap.org/claims/EmailAddress">
    <fed:DisplayName>Email Address</fed:DisplayName>
    <fed:Description>Email Address description</fed:Description>
  </fed:ClaimType>
</fed:UriNamedClaimTypesOffered>
```

Authorization Common Claims Dialect Example

```
<!-- Example shows use of a EmailAddress claim in auth claim type -->
<wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2006/12/authorization/authclaims">
  <auth:ClaimType Uri="http://schemas.xmlsoap.org/claims/EmailAddress"/>
</wst:Claims>
```

SAML 1.1 Example

EmailAddress claim type can be used in the Subject/NameIdentifier element of an AuthenticationStatement or AttributeStatement. The value of the Format attribute SHOULD be URI: urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

```
<!-- Example illustrates use of an EmailAddress claim in
Subject/NameIdentifier element of an AuthenticationStatement that is part of
a SAML assertion (not shown here) -->
<AuthenticationStatement
  AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
  AuthenticationInstant="2003-05-12T19:29:23">
  <Subject>
    <NameIdentifier
      Format="urn:oasis:names:tc:SAML:1.1nameid-format:emailAddress">
      johnd@fabrikam.com
    </NameIdentifier>
  </Subject>
</AuthenticationStatement>
```

EmailAddress claim type can be used in the Attribute element of an AttributeStatement. The value of the AttributeNamespace attribute SHOULD be URL: http://schemas.xmlsoap.org/claims. The value for AttributeName MUST be “EmailAddress”.

```
<!-- Example illustrates use of an EmailAddress claim in an Attribute element
of an AttributeStatment that is part of a SAML assertion (not shown here) -->
<AttributeStatement>
  <Attribute
    AttributeName="EmailAddress"
    AttributeNamespace="http://schemas.xmlsoap.org/claims">
    <AttributeValue>
      johnd@fabrikam.com
    </AttributeValue>
  </Attribute>
</AttributeStatement>
```

User Principal Name (UPN)

UPN claim type is used to identify a specific security principal via a User Principal Name. The value MUST be unique such that it can be used for identification and authorization.

Offered Claim Type Example

```
<!-- Example shows use of a UPN claim in Offered Claim type -->
<fed:UriNamedClaimTypesOffered>
  <fed:ClaimType Uri="http://schemas.xmlsoap.org/claims/UPN">
    <fed:DisplayName>User Principal Name</fed:DisplayName>
    <fed:Description>User Principal Name description</fed:Description>
  </fed:ClaimType>
</fed:UriNamedClaimTypesOffered>
```

Authorization Common Claims Dialect Example

```
<!-- Example shows use of a UPN claim in auth claim type -->
<wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2006/12/authorization/authclaims">
  <auth:ClaimType Uri="http://schemas.xmlsoap.org/claims/UPN"/>
</wst:Claims>
```

SAML 1.1 Example

UPN claim type can be used in the Subject/NameIdentifier element of an AuthenticationStatement or AttributeStatement. The value of the Format attribute SHOULD be URI: http://schemas.xmlsoap.org/claims/UPN

```
<!-- The example below illustrates the use of an UPN claim in
Subject/NameIdentifier element of an AuthenticationStatement that is part of
a SAML assertion (not shown here) -->
<AuthenticationStatement
  AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
  AuthenticationInstant="2003-05-12T19:29:23">
  <Subject>
    <NameIdentifier Format="http://schemas.xmlsoap.org/claims/UPN">
      johndoe@humanresource.fabrikam.com
    </NameIdentifier>
  </Subject>
</AuthenticationStatement>
```

UPN claim type can be used in the Attribute element of an AttributeStatement. The value of the AttributeNamespace attribute SHOULD be URL: http://schemas.xmlsoap.org/claims. The value for AttributeName MUST be “UPN”.

```
<!-- This example illustrates use of an UPN claim in Attribute element of an
AttributeStatment that is part of a SAML assertion (not shown here) -->
<AttributeStatement>
  <Attribute
    AttributeName="UPN"
    AttributeNamespace="http://schemas.xmlsoap.org/claims">
    <AttributeValue>
      johndoe@humanresource.fabrikam.com
    </AttributeValue>
  </Attribute>
</AttributeStatement>
```

Common Name

CommonName claim type is used to identify a security principal via a CN value consistent with X.500 naming conventions. The value of this claim is not necessarily unique and SHOULD NOT be used for authorization purposes. It is suitable for displaying a friendly name for personalization.

Offered Claim Type Example

```
<!-- Example shows use of a CommonName claim in Offered Claim type -->
<fed:UriNamedClaimTypesOffered>
  <fed:ClaimType Uri="http://schemas.xmlsoap.org/claims/CommonName">
    <fed:DisplayName>Common Name</fed:DisplayName>
    <fed:Description>Common Name description</fed:Description>
  </fed:ClaimType>
</fed:UriNamedClaimTypesOffered>
```

Authorization Common Claims Dialect Example

```
<!-- Example shows use of a CommonName claim in auth claim type -->
<wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2006/12/authorization/authclaims">
```

```
  <auth:ClaimType Uri="http://schemas.xmlsoap.org/claims/CommonName"/>
</wst:Claims>
```

SAML 1.1 Example

CommonName claim type can be used in the Subject/NameIdentifier element of an AuthenticationStatement or AttributeStatement. The value of the Format attribute SHOULD be URI: http://schemas.xmlsoap.org/claims/CommonName

```
<!-- Example illustrates use of a CommonName claim in Subject/NameIdentifier
element of an AuthenticationStatement that is part of a SAML assertion (not
shown here) -->
<AuthenticationStatement
  AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
  AuthenticationInstant="2003-05-12T19:29:23">
  <Subject>
    <NameIdentifier Format="http://schemas.xmlsoap.org/claims/CommonName">
      john doe
    </NameIdentifier>
  </Subject>
</AuthenticationStatement>
```

CommonName claim type can be used in the Attribute element of an AttributeStatement. The value of the AttributeNamespace attribute SHOULD be URL: http://schemas.xmlsoap.org/claims. The value for AttributeName MUST be “CommonName”.

```
<!-- Example illustrates use of a CommonName claim in Attribute element of an
AttributeStatment that is part of a SAML assertion (not shown here) -->
<AttributeStatement>
  <Attribute
    AttributeName="CommonName"
    AttributeNamespace="http://schemas.xmlsoap.org/claims/CommonName">
    <AttributeValue>
      John Doe
    </AttributeValue>
  </Attribute>
</AttributeStatement>
```

Group Claim Type

Group claim type is used to indicate the association of the subject with other security principals. Interpretation of that association is application specific, but is typically treated as “group or role membership.”

Offered Claim Type Example

```
<!-- Example shows use of a Group claim in Offered Claim type -->
<fed:UriNamedClaimTypesOffered>
  <fed:ClaimType Uri="http://schemas.xmlsoap.org/claims/Group">
    <fed:DisplayName>Group</fed:DisplayName>
    <fed:Description>Group description</fed:Description>
  </fed:ClaimType>
</fed:UriNamedClaimTypesOffered>
```

Authorization Common Claims Dialect Example

```
<!-- Example shows use of a Group claim in auth claim type -->
<wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2006/12/authorization/authclaims">
  <auth:ClaimType Uri="http://schemas.xmlsoap.org/claims/Group"/>
</wst:Claims>
```

SAML 1.1 Example

Group claim type MUST NOT be used in the *Subject/NameIdentifier* element.

Group claim type can only be used in the *Attribute* element of an *AttributeStatement*. The value of the *AttributeNamespace* attribute SHOULD be URL: http://schemas.xmlsoap.org/claims/Group. The value for *AttributeName* MUST be "Group".

```
<!-- Example illustrates use of a Group claim in Attribute element of an
AttributeStatment that is part of a SAML assertion (not shown here) -->
<AttributeStatement>
  <Attribute
    AttributeName="Group"
    AttributeNamespace="http://schemas.xmlsoap.org/claims">
    <AttributeValue>
      AccountManagers
    </AttributeValue>
  </Attribute>
</AttributeStatement>
```

NameValue Claim Type

NameValue claim type is an extensibility point to enable applications to define their name/value claims.

Offered Claim Type Example

```
<!-- Example shows use of a NameValue claim in Offered Claim type -->
<fed:UriNamedClaimTypesOffered>
  <fed:ClaimType Uri="http://schemas.xmlsoap.org/claims/NameValue/Any">
    <fed:DisplayName>Any</fed:DisplayName>
    <fed:Description>Specific to offered Name/Value</fed:Description>
  </fed:ClaimType>
</fed:UriNamedClaimTypesOffered>
```

Authorization Common Claims Dialect Example

```
<!-- Example shows use of a NameValue claim in auth claim type -->
<wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2006/12/authorization/authclaims">
  <auth:ClaimType Uri="http://schemas.xmlsoap.org/claims/NameValue/Any"/>
</wst:Claims>
```

SAML 1.1 Example

NameValue claim type MUST NOT be used in the *Subject/NameIdentifier* element.

NameValue claim type can only be used in the *Attribute* element of an *AttributeStatement*. The value of the *AttributeNamespace* attribute SHOULD be URL: http://schemas.xmlsoap.org/claims. The value for *AttributeName* is app specific.

```
<!-- Example illustrates use of a NameValue claim in Attribute element of an
AttributeStatment that is part of a SAML assertion (not shown here) -->
<AttributeStatement>
  <Attribute
   AttributeName="projectName"
   AttributeNamespace="http://schemas.xmlsoap.org/claims">
    <AttributeValue>
      rocketV
    </AttributeValue>
  </Attribute>
</AttributeStatement>
```

Appendix B – Acknowledgements

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

Greg Carpenter, Microsoft Corporation

Colleen Evans, Microsoft Corporation

Heather Hinton, IBM

Chris Kaler, Microsoft Corporation

Arun Nanda, Microsoft Corporation

Nataraj Nagaratnam, IBM

Roberto Ruggeri, Microsoft Corporation

Jorgen Thelin, Microsoft Corporation

Appendix C – XML Namespaces

The following table lists XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

Prefix	Namespace	Specification(s)
fed	http://schemas.xmlsoap.org/ws/2006/12/federation	[WS-Federation]
auth	http://schemas.xmlsoap.org/ws/2006/12/authorization	[WS-Federation]
priv	http://schemas.xmlsoap.org/ws/2006/12/privacy	[WS-Federation]
mex	http://schemas.xmlsoap.org/ws/2004/09/mex	[WS-MetadataExchange]
wst	http://docs.oasis-open.org/ws-sx/ws-trust/200512	[WS-Trust]
sp	http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702	[WS-SecurityPolicy]
wsp	http://schemas.xmlsoap.org/ws/2004/09/policy	[WS-Policy, WS-PolicyAttachment]
s	http://schemas.xmlsoap.org/soap/envelope/ or http://www.w3.org/2003/05/soap-envelope	[SOAP]
wsa	http://www.w3.org/2005/08/addressing	[WS-Addressing]
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	[WSS]
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	[WSS]
ds	http://www.w3.org/2000/09/xmldsig#	[XML-Signature]

Appendix D – References

[WS-Federation]	"Web Services Federation Language (WS-Federation)", December 2006 http://specs.xmlsoap.org/ws/2006/12/federation/
[WS-MetadataExchange]	"Web Services Metadata Exchange (WS-MetadataExchange)", August 2006 http://schemas.xmlsoap.org/ws/2004/09/mex/
[WS-Trust]	OASIS Standard, "WS-Trust 1.3", March 2007 http://docs.oasis-open.org/ws-sx/ws-trust/200512
[WS-Policy]	W3C Member Submission "Web Services Policy 1.2 - Framework", 25 April 2006. http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/
[WS-PolicyAttachment]	W3C Member Submission "Web Services Policy 1.2 - Attachment", 25 April 2006. http://www.w3.org/Submission/2006/SUBM-WS-PolicyAttachment-20060425/
[SOAP]	W3C Note, "SOAP: Simple Object Access Protocol 1.1", 08 May 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
	W3C Recommendation, "SOAP 1.2 Part 1: Messaging Framework", 24 June 2003. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/
[WS-Addressing]	W3C Recommendation, "Web Services Addressing (WS-Addressing)", 9 May 2006. http://www.w3.org/TR/2006/REC-ws-addr-core-20060509
[WSS]	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)", March 2004. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
	OASIS Standard, "OASIS Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)", February 2006. http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[XML-Signature]	W3C Recommendation, "XML-Signature Syntax and Processing", 12 February 2002. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
[XML-Encrypt]	W3C Recommendation, "XML Encryption Syntax and Processing", 10 December 2002. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
[HTTP]	IETF Standard, "Hypertext Transfer Protocol -- HTTP/1.1" January 1997 http://www.ietf.org/rfc/rfc2068.txt
[HTTPS]	IETF Standard, "The TLS Protocol", January 1999. http://www.ietf.org/rfc/rfc2246.txt