Web Services Dynamic Discovery (WS-Discovery)

February 2004

Co-Developers

John Beatty, BEA Systems
Gopal Kakivaya, Microsoft
Devon Kemp, Canon
Brad Lovering, Microsoft
Bryan Roe, Intel
Jeffrey Schlimmer (Editor), Microsoft
Guillaume Simonnet, Microsoft
Jack Weast, Intel

Copyright Notice

Abstract

This specification defines a multicast discovery protocol to locate services. By default, probes are sent to a multicast group, and target services that match return a response directly to the requester. To scale to a large number of endpoints, the protocol defines the multicast suppression behavior if a discovery proxy is available on the network. To minimize the need for polling, target services that wish to be discovered send an announcement when they join and leave the network.

Composable Architecture

The Web service specifications (WS-*) are designed to be composed with each other to provide a rich set of tools to provide security in the Web services environment. This specification specifically relies on other Web service specifications to provide secure, reliable, and/or transacted message delivery and to express Web service and client policy.

Status

WS-Discovery and related specifications are provided for use as-is and for review and evaluation only. Microsoft, BEA, Canon and Intel will solicit your contributions and suggestions in the near future. Microsoft, BEA, Canon and Intel make no warrantees or representations regarding the specification in any manner whatsoever.

Table of Contents

1. Introduction

This specification defines a multicast discovery protocol to locate services. The primary mode of discovery is a client searching for one or more target services. To find a target service by the type of the target service, a scope in which the target service resides, or both, a client sends a probe message to a multicast group; target services that match the probe send a response directly to the client. To locate a target service by name, a client sends a resolution request message to the same multicast group, and again, the target service that matches sends a response directly to the client.

To minimize the need for polling, when a target service joins the network, it sends an announcement message to the same multicast group. By listening to this multicast group, clients can detect newly-available target services without repeated probing.

To scale to a large number of endpoints, this specification defines multicast suppression behavior if a discovery proxy is available on the network. Specifically, when a discovery proxy detects a probe or resolution request sent by multicast, the discovery proxy sends an announcement for itself. By listening for these announcements, clients detect discovery proxies and switch to use a discovery proxy-specific protocol. However, if a discovery proxy is unresponsive, clients revert to use the protocol described herein. While the definition of a discovery proxy-specific protocol is beyond the scope of this specification, it is expected that any such protocol would define search messages clients send directly to the discovery proxy rather than to a multicast group.

To support managed networks, this specification acknowledges that clients and/or target services may be configured to behave differently than defined herein. For example, another specification may define a well-known DHCP record containing the address of a discovery proxy, and compliance with that specification may require endpoints to send messages to this discovery proxy rather than to a multicast group. While the specific means of such configuration is beyond the scope of this specification, it is expected that any such configuration would allow clients and/or target services to migrate smoothly between carefully-managed and ad hoc networks.

1.1 Requirements

This specification intends to meet the following requirements:

- Allow discovery of services in ad hoc networks with a minimum of networking services (e.g., no DNS or directory services).

- Leverage network services to reduce network traffic in managed networks where such services exist.
- Enable smooth transitions between ad hoc and managed networks.
- Enable discovery of resource-limited service implementations.
- Support bootstrapping to other Web service protocols as well as other transports.
- Enable discovery of services by type and within scope.
- Leverage other Web service specifications for secure, reliable, transacted message delivery.
- Provide extensibility for more sophisticated and/or currently unanticipated scenarios.
- Support both SOAP 1.1 (see [SOAP 1.1]) and SOAP 1.2 (see [SOAP 1.2]) Envelopes.

1.2 Non-Requirements

This specification does not intend to meet the following requirements:

- Provide liveness information on services.
- Define a data model for service description or define rich queries over that description.
- Support Internet-scale discovery.

1.3 Example

Table 1 lists an example Probe message multicast by a Client searching for a printer.

Table 1: Example Probe

```
(01) <s:Envelope
(02)     xmlns:a='http://schemas.xmlsoap.org/ws/2003/03/addressing'
(03)     xmlns:d='http://schemas.xmlsoap.org/ws/2004/02/discovery'
(04)     xmlns:i='http://printer.example.org/2003/imaging'
(05)     xmlns:s='http://www.w3.org/2003/05/soap-envelope' >
(06)   <s:Header>
(07)     <a:Action>
(08)       http://schemas.xmlsoap.org/ws/2004/02/discovery/Probe
(09)     </a:Action>
(10)     <a:MessageID>
(11)       uuid:0a6dc791-2be6-4991-9af1-454778a1917a
(12)     </a:MessageID>
(13)     <a:To>http://schemas.xmlsoap.org/ws/2004/02/discovery</a:To>
(14)   </s:Header>
(15)   <s:Body>
(16)     <d:Probe>
(17)       <d:Types>i:PrintBasic</d:Types>
```

```
(18)       <d:Scope
(19)   MatchBy='http://schemas.xmlsoap.org/ws/2004/02/discovery/ldap'>
(20)         ldap:///ou=engineering,o=examplecom,c=us
(21)       </d:Scope>
(22)     </d:Probe>
(23)   </s:Body>
(24) </s:Envelope>
```

Lines (07-09) in Table 1 indicate the message is a Probe, and Line (13) indicates it is being sent to a well-known address.

Because there is no explicit ReplyTo SOAP header block (see [WS-Addressing]), any response to this Probe will be sent as a UDP packet to the source IP address and port of the Probe transport header.

Lines (17-21) specify two constraints on the Probe: Line (17) constrains responses to Target Services that implement a basic print Type; Lines (18-21) constrain responses to Target Services in the Scope for an engineering department. Only Target Services that satisfy both of these constraints will respond. Though both constraints are included in this example, a Probe is not required to include either.

Table 2 lists an example Probe Match message sent in response to the Probe in Table 1.

Table 2: Example Probe Match

```
(01) <s:Envelope
(02)   xmlns:a='http://schemas.xmlsoap.org/ws/2003/03/addressing'
(03)   xmlns:d='http://schemas.xmlsoap.org/ws/2004/02/discovery'
(04)   xmlns:i='http://printer.example.org/2003/imaging'
(05)   xmlns:p='http://schemas.xmlsoap.org/ws/2002/12/policy'
(06)   xmlns:s='http://www.w3.org/2003/05/soap-envelope' >
(07)   <s:Header>
(08)     <a:Action>
(09)       http://schemas.xmlsoap.org/ws/2004/02/discovery/ProbeMatch
(10)     </a:Action>
(11)     <a:MessageID>
(12)       uuid:e32e6863-ea5e-4ee4-997e-69539d1ff2cc
(13)     </a:MessageID>
(14)     <a:RelatesTo>
(15)       uuid:0a6dc791-2be6-4991-9af1-454778a1917a
(16)     </a:RelatesTo>
(17)     <a:To>
(18)    http://schemas.xmlsoap.org/ws/2003/03/addressing/role/anonymous
```

```
(19)      </a:To>
(20)    </s:Header>
(21)    <s:Body>
(22)      <d:ProbeMatch>
(23)        <a:EndpointReference>
(24)          <a:Address>
(25)            uuid:98190dc2-0890-4ef8-ac9a-5940995e6119
(26)          </a:Address>
(27)          <p:Policy>
(28)            <d:SoapHttpRequestReplyAddress>
(29)              http://prn-example/PRN42/b42-1668-a
(30)            </d:SoapHttpRequestReplyAddress>
(31)          </p:Policy>
(32)        </a:EndpointReference>
(33)        <d:Types>i:PrintBasic i:PrintAdvanced</d:Types>
(34)        <d:Scopes>
(35)          ldap:///ou=engineering,o=examplecom,c=us
(36)          ldap:///ou=floor1,ou=b42,ou=anytown,o=examplecom,c=us
(37)        </d:Scopes>
(38)        <d:MetadataVersion>75965</d:MetadataVersion>
(39)      </d:ProbeMatch>
(40)    </s:Body>
(41)  </s:Envelope>
```

Lines (08-10) in Table 2 indicate this message is a Probe Match, and Lines (14-16) indicate that it is a response to the Probe in Table 1. Because the Probe did not have an explicit ReplyTo SOAP header block, Lines (17-19) indicate that the response was sent to the source IP address and port of the transport header of the Probe.

Lines (22-39) describe a single Target Service.

Lines (24-26) contain the stable, unique identifier for the Target Service that is constant across network interfaces, addresses, and IPv4/v6. In this case, the value is a UUID scheme URI, but it may be a transport URI (like the one in Line 29) if it meets stability and uniqueness requirements. Generally, the stable, unique identifier for a Target Service includes not only the Address header block but also the ReferenceProperties header block [WS-Addressing]; in this example, the Target Service is identified only by the Address header block.

Lines (27-31) include just enough policy (see [WS-Policy]) to contact the Target Service. Specifically, Lines (28-30) indicate the transport addresses where the Target Service may be reached; in this case, Line (29) indicates a single HTTP transport address that accepts SOAP messages; as in this case, unless explicitly indicated otherwise, the transport understands SOAP 1.1 (see [SOAP 1.1]) messages.

Line (33) lists the Types (see, e.g., [WSDL 1.1]) implemented by the Target Service, in this example, a basic print type that matched theiy cytgmr4e3b Probe as well as an advanced print type.

Lines (34-37) list two administrative Scopes, one that matched the Probe as well as another that is specific to a particular physical location.

Line (38) contains the version of the metadata for the Target Service; as explained below, this version is incremented if there is a change in the metadata for the Target Service (including Lines 27-31 and Lines 33-37).

2. Terminology and Notation

2.1 Terminology

Target Service
: An endpoint that makes itself available for discovery.

Client
: An endpoint that searches for Target Service(s).

Discovery Proxy
: An endpoint that facilitates discovery by Clients among a large number of endpoints. Discovery Proxies are an optional component of the architecture.

Hello
: A message sent by a Target Service when it joins a network; the message contains key information for the Target Service.

Bye
: A best-effort message sent by a Target Service when it leaves a network.

Probe
: A message sent by a Client searching for a Target Service by Type and/or Scope.

Resolve
: A message sent by a Client searching for a Target Service by name.

Type
: An identifier for a set of messages an endpoint sends and/or receives (e.g., a WSDL 1.1 portType, see [WSDL 1.1]).

Scope
: An extensibility point that may be used to organize Target Services into logical groups.

Metadata
: Information about the Target Service; includes, but is not limited to, network addresses where a Target Service may be reached, transports and protocols it understands, Types it implements, and Scopes it is in.

2.2 Notational Conventions

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

This specification uses the following syntax to define normative outlines for messages:

The syntax appears as an XML instance, but values in italics indicate data types instead of values.

Characters are appended to elements and attributes to indicate cardinality:

- "?" (0 or 1)
- "*" (0 or more)
- "+" (1 or more)
- The character "|" is used to indicate a choice between alternatives.
- The characters "[" and "]" are used to indicate that contained items are to be treated as a group with respect to cardinality or choice.

Ellipses (i.e., "...") indicate a point of extensibility that allows other child or attribute content. Additional children and/or attributes MAY be added at the indicated extension points but MUST NOT contradict the semantics of the parent and/or owner, respectively. If a receiver does not recognize an extension, the receiver SHOULD ignore the extension.

XML namespace prefixes (see Table 3) are used to indicate the namespace of the element being defined.

2.3 XML Namespaces

The XML Namespace URI that MUST be used by implementations of this specification is:

```
http://schemas.xmlsoap.org/ws/2004/02/discovery
```

Table 3 lists XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

Table 3: Prefixes and XML Namespaces used in this specification

Prefix	XML Namespace	Specification(s)
s	(Either SOAP 1.1 or 1.2)	(Either SOAP 1.1 or 1.2)
s11	http://schemas.xmlsoap.org/soap/envelope/	(see [SOAP 1.1])
s12	http://www.w3.org/2003/05/soap-envelope	(see [SOAP 1.2])
a	http://schemas.xmlsoap.org/ws/2003/03/addressing	(see [WS-Addressing])
d	http://schemas.xmlsoap.org/ws/2004/02/discovery	This specification
p	http://schemas.xmlsoap.org/ws/2002/12/policy	(see [WS-Policy])
xs	http://www.w3.org/2001/XMLSchema	(see [XML Schema Part 1, 2])

2.4 Protocol Assignments

Multicast messages described herein MUST be sent using the following assignments:

- PORT: 3702 (see [IANA])
- IPv4 multicast address: 239.255.255.250
- IPv6 multicast address: FF02::C (link-local scope)

Messages sent over UDP MUST be sent using SOAP over UDP (see [SOAP/UDP]). To compensate for possible UDP unreliability, senders MUST use the example transmission algorithm in Appendix A of SOAP over UDP; Table 4 revises one default value for a parameter within that algorithm.

Table 4: Revised default value for a SOAP over UDP transmission parameter

Parameter	Default Value
UNICAST_UDP_REPEAT	2

As designated below, before sending some message types defined herein, the sender MUST wait for a timer before sending the message. This timer MUST be set to a random value between 0 and APP_MAX_DELAY. Table 5 specifies the default value for this parameter.

Table 5: Default value for an application-level transmission parameter

Parameter	Default Value
APP_MAX_DELAY	500 milliseconds

The default values in Table 4 and Table 5 MAY be revised by other specifications.

Note: The authors expect these parameters to be adjusted based on interoperability test results.

2.5 Compliance

An endpoint MAY implement more than one of Target Service, Discovery Proxy, and Client; however, for each implemented, it MUST implement them as specified herein.

An implementation is not compliant with this specification if it fails to satisfy one or more of the MUST or REQUIRED level requirements defined herein for the roles it implements.

3. Model

Figure 1 depicts the message exchanges between a Target Service and a Client.

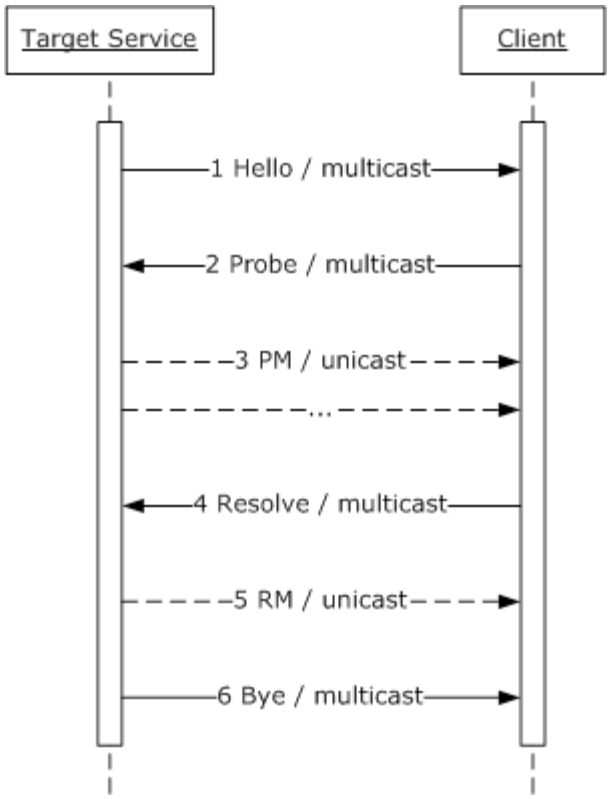

Figure 1: Message exchanges.

Starting on the left of Figure 1, initially a Target Service (1) sends a multicast Hello when it joins a network. A Target Service may (2) receive multicast a Probe at any time and (3) sends a unicast Probe Match (PM) if the Target Service matches a Probe. Similarly, a Target Service may (4) receive multicast a Resolve at any time and (5) sends a unicast Resolve Match (RM) if it is the target of a Resolve. Conceptually, the Hello, Probe Match, and Resolve Match messages carry the same kind of information; specific differences are detailed below. Finally, when a Target Service leaves a network, it makes an effort to (6) send a multicast Bye.

Moving to the right of Figure 1, a Client mirrors Target Service messages. A Client listens to multicast Hello, may Probe to find Target Services or may Resolve to find a particular Target Service, and listens to multicast Bye.

To limit multicast traffic, Clients operate in one of two modes as depicted in Figure 2.

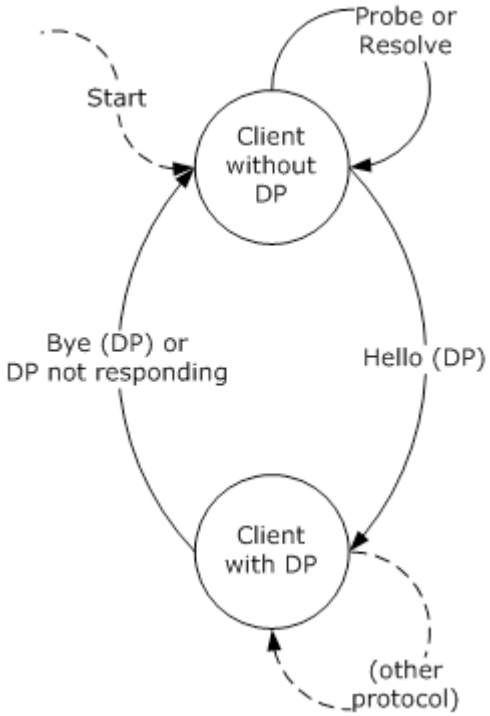

Figure 2: Client states.

By default, a new Client assumes that no Discovery Proxy (DP) is available, listens for announcements, sends Probe and/or Resolve messages, and listens for Probe Match and/or Resolve Match messages as specified herein.

However, if one or more DP are available, they send a unicast Hello with a well-known "discovery proxy" type (described below) in response to any Probe or Resolve. Clients listen for this signal that one or more DP are available, and for subsequent searches, Clients use a discovery proxy-specific protocol to communicate with one or more of them. While the definition of such a protocol is beyond the scope of this specification, it is expected that any such protocol would define search messages clients send directly to the discovery proxy rather than to a multicast group. If these DP are unresponsive, or if they send a Bye, Clients revert to using the messages specified herein.

This design minimizes discovery latency in ad hoc networks without increasing multicast traffic in managed networks. To see this, note that a Client only generates multicast traffic when it sends a Probe or Resolve; while a Client could Probe (or Resolve) for a DP *before* Probing (or Resolving) for a Target Service of interest, this is just as expensive in a managed network (in terms of network traffic) as allowing the Client to Probe (or Resolve) for the Target Service directly and having the DP respond to signal its presence; the reduced latency in ad hoc networks arises because the Client does not need to explicitly search and wait for possible DP responses.

Unlike a Client, a Target Service always sends (multicast) Hello and Bye, and always responds to Probe and Resolve with (unicast) Probe Match and Resolve Match, respectively. It does not need to explicitly recognize and/or track the availability of a Discovery Proxy. The Hello and Bye are too infrequent and therefore generate too little multicast traffic to warrant adding complexity to Target Service behavior.

4. Hello and Bye

Messages described in this section MUST be implemented by a Target Service, MUST be implemented by a Discovery Proxy (for itself, not for other Target Services), and MAY be implemented by a Client.

4.1 Hello

A Target Service MUST send a one-way Hello when any of the following occur:

- It joins a network. This may be detected through low-level mechanisms, such as wireless beacons, or through a change in IP connectivity on one or more of its network interfaces.
- Its metadata changes (see /s:Envelope/s:Body/*/d:MetadataVersion below).

To minimize the risk of a network storm (e.g., after a network crash and recovery or power black out and restoration), a Target Service MUST wait for a timer after one of the above occurs before sending the Hello as described in Section 2.4 Protocol Assignments.

Additionally, a Discovery Proxy MUST send a unicast Hello in response to any Probe or any Resolve (see Section 5. Probe and Probe Match and Section 6. Resolve and Resolve Match, respectively). The meaning of this message is that the Client that sent the Probe or Resolve SHOULD switch to a discovery proxy-specific protocol (see Section 3. Model).

The Hello MUST be sent to the IPv4 and/or IPv6 multicast addresses listed in Section 2.4 Protocol Assignments.

The normative outline for Hello is:

```
<s:Envelope ...>
  <s:Header ...>
    <a:Action ...>
        http://schemas.xmlsoap.org/ws/2004/02/discovery/Hello
    </a:Action>
    <a:MessageID ...>xs:anyURI</a:MessageID>
    <a:To ...>http://schemas.xmlsoap.org/ws/2004/02/discovery</a:To>
    <d:AppSequence ... />
    ...
  </s:Header>
  <s:Body ...>
    <d:Hello>
      <a:EndpointReference ...>
        <a:Address ...>xs:anyURI</a:Address>
       [<a:ReferenceProperties ...>...</a:ReferenceProperties>]?
        ...
       [<p:Policy>policy expression</p:Policy>]?
      </a:EndpointReference>
```

```
      [<d:Types ...>list of xs:QName</d:Types>]?
      [<d:Scopes ...>list of xs:anyURI</d:Scopes>]?
       <d:MetadataVersion ...>xs:nonNegativeInteger</d:MetadataVersion>
       ...
     </d:Hello>
  </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/a:Action
: If a SOAP Action URI is used in the binding for SOAP, the value indicated herein MUST be used for that URI.

/s:Envelope/s:Header/d:AppSequence
: MUST be included to allow ordering Hello and Bye messages. See definition in Appendix I – Application Sequencing.

/s:Envelope/s:Body/*/a:EndpointReference
: MUST include enough metadata to bootstrap communication with the Target Service, and SHOULD NOT include more.

/s:Envelope/s:Body/*/a:EndpointReference/a:Address
: MAY be resolvable to a transport and network address.

/s:Envelope/s:Body/*/a:EndpointReference/a:Address and
/s:Envelope/s:Body/*/a:EndpointReference/a:ReferenceProperties
: Per WS-Addressing [WS-Addressing], the combination of these MUST be a stable, globally-unique identifier for the Target Service; MUST be constant across network interfaces, IP addresses, and IPv4/v6; MUST be persistent across re-initialization and changes in the metadata of this Target Service.

/s:Envelope/s:Body/*/a:EndpointReference/p:Policy
: MAY include instances of policy assertions [WS-Policy] to identify transport addresses that MAY be used to communicate with the Target Service. (See Appendix II – Provisional Policy Assertions.) There is no requirement to explicitly identify transport addresses for the messages defined herein (see 2.4 Protocol Assignments); however, a Target Service MAY support the messages defined herein over the transport addresses explicitly identified.

/s:Envelope/s:Body/*/d:Types
: Unordered set of Types implemented by the Target Service or Discovery Proxy. Implicitly includes d:Discovery and d:TargetService (see Appendix IV – WSDL); other Types MAY be implied by compliance with other specifications.
  - For a Target Service, if omitted, implied value is a set that includes d:Discovery and d:TargetService.
  - For a Discovery Proxy, MUST NOT be omitted and MUST explicitly include d:DiscoveryProxy.

/s:Envelope/s:Body/*/d:Scopes
: Unordered set of Scopes the Target Service is in. If included, MUST be a set of absolute URIs. If omitted, implied value is a set that includes 'http://schemas.xmlsoap.org/ws/2004/02/discovery/adhoc'.

/s:Envelope/s:Body/*/d:MetadataVersion

Incremented by >= 1 whenever there is a change in the metadata of the Target Service. Metadata includes, but is not limited to, ../a:EndpointReference/p:Policy, ../d:Types, and ../d:Scopes. By design, this value MAY be used for cache control of Target Service metadata.

To minimize the need to Probe, Clients SHOULD listen for Hello messages and store (or update) information for the corresponding Target Service. Note that a Target Service MAY vary the amount of metadata it includes in Hello messages (or Probe Match or Resolve Match messages), and consequently, a Client may receive two such messages containing the same /s:Envelope/s:Body/*/d:MetadataVersion but containing different metadata. If a Client chooses to cache metadata, it MAY, but is not constrained to, adopt any of the following behaviors:

- Cache the union of the previously cached and new metadata.
- Replace the previously cached with new metadata.
- Use some other means to retrieve more complete metadata.

However, to prevent network storms, a Client SHOULD NOT delete cached metadata and repeat a Probe or Resolve if it detects differences in contained metadata.

Table 6 lists an example Hello for the same Target Service that responded with a Probe Match in Table 2.

Table 6: Example Hello

```
(01) <s:Envelope
(02)   xmlns:a='http://schemas.xmlsoap.org/ws/2003/03/addressing'
(03)   xmlns:d='http://schemas.xmlsoap.org/ws/2004/02/discovery'
(04)   xmlns:i='http://printer.example.org/2003/imaging'
(05)   xmlns:p='http://schemas.xmlsoap.org/ws/2002/12/policy'
(06)   xmlns:s='http://www.w3.org/2003/05/soap-envelope' >
(07)   <s:Header>
(08)     <a:Action>
(09)         http://schemas.xmlsoap.org/ws/2004/02/discovery/Hello
(10)     </a:Action>
(11)     <a:MessageID>
(12)         uuid:0a6dc791-2be6-4991-9af1-454778a1917a
(13)     </a:MessageID>
(14)     <a:To>http://schemas.xmlsoap.org/ws/2004/02/discovery</a:To>
(15)     <d:AppSequence InstanceId='1077004800' MessageNumber='1' />
(16)   </s:Header>
(17)   <s:Body>
(18)     <d:Hello>
(19)       <a:EndpointReference>
(20)         <a:Address>
```

```
(21)           uuid:98190dc2-0890-4ef8-ac9a-5940995e6119
(22)         </a:Address>
(23)         <p:Policy>
(24)           <d:SoapHttpRequestReplyAddress>
(25)             http://prn-example/PRN42/b42-1668-a
(26)           </d:SoapHttpRequestReplyAddress>
(27)         </p:Policy>
(28)       </a:EndpointReference>
(29)       <d:Types>i:PrintBasic i:PrintAdvanced</d:Types>
(30)       <d:Scopes>
(31)         ldap:///ou=engineering,o=examplecom,c=us
(32)         ldap:///ou=floor1,ou=b42,ou=anytown,o=examplecom,c=us
(33)       </d:Scopes>
(34)       <d:MetadataVersion>75965</d:MetadataVersion>
(35)     </d:Hello>
(36)   </s:Body>
(37) </s:Envelope>
```

Lines (08-10) indicate this is a Hello, and because Line (14) is set to the distinguished URI defined herein, this is a multicast Hello. Line (15) contains an instance identifier as well as a message number; this information allows the receiver to order Hello and Bye messages from a single sender. Lines (19-34) are identical to the corresponding lines in the Probe Match in Table 2.

4.2 Bye

A Target Service SHOULD send a one-way Bye message when it is preparing to leave a network. (A Target Service MUST NOT send a Bye message when its metadata changes.)

A Target Service MAY send the Bye without waiting for a timer.

The Bye MUST be sent to the IPv4 and/or IPv6 multicast addresses listed in Section 2.4 Protocol Assignments.

The normative outline for Bye is:

```
<s:Envelope ...>
  <s:Header ...>
    <a:Action ...>
        http://schemas.xmlsoap.org/ws/2004/02/discovery/Bye
    </a:Action>
    <a:MessageID ...>xs:anyURI</a:MessageID>
    <a:To ...>http://schemas.xmlsoap.org/ws/2004/02/discovery</a:To>
    <d:AppSequence ... />
```

```
    ...
  </s:Header>
  <s:Body ...>
    <d:Bye ...>
      <a:EndpointReference ...>
        <a:Address ...>xs:anyURI</a:Address>
       [<a:ReferenceProperties ...>...</a:ReferenceProperties>]?
        ...
      </a:EndpointReference>
      ...
    </d:Bye>
  </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/a:Action
: If a SOAP Action URI is used in the binding for SOAP, the value indicated herein MUST be used for that URI.

/s:Envelope/s:Body/*/a:EndpointReference/a:Address and
/s:Envelope/s:Body/*/a:EndpointReference/a:ReferenceProperties
: The combination of these MUST be a stable, globally-unique identifier for the Target Service (see Section 4.1 Hello for details).

Clients SHOULD listen for Bye messages, marking or removing corresponding information as invalid. Clients MAY wish to retain information associated with a Target Service that has left the network, for instance if the Client expects the Target Service to rejoin the network at some point in the future. Conversely, Clients MAY discard information associated with a Target Service at any time, based on, for instance, preset maximums on the amount of memory allocated for this use, lack of communication to the Target Service, preferences for other Target Service Types or scopes, and/or other application-specific preferences.

Table 7 lists an example Bye message corresponding to the Hello in Table 6.

Table 7: Example Bye

```
(01) <s:Envelope
(02)     xmlns:a='http://schemas.xmlsoap.org/ws/2003/03/addressing'
(03)     xmlns:d='http://schemas.xmlsoap.org/ws/2004/02/discovery'
(04)     xmlns:s='http://www.w3.org/2003/05/soap-envelope' >
(05)   <s:Header>
(06)     <a:Action>
(07)         http://schemas.xmlsoap.org/ws/2004/02/discovery/Bye
(08)     </a:Action>
(09)     <a:MessageID>
```

```
(10)          uuid:337497fa-3b10-43a5-95c2-186461d72c9e
(11)      </a:MessageID>
(12)      <a:To>http://schemas.xmlsoap.org/ws/2004/02/discovery</a:To>
(13)      <d:AppSequence InstanceId='1077004800' MessageNumber='2' />
(14)    </s:Header>
(15)    <s:Body>
(16)      <d:Bye>
(17)        <a:EndpointReference>
(18)          <a:Address>
(19)            uuid:98190dc2-0890-4ef8-ac9a-5940995e6119
(20)          </a:Address>
(21)        </a:EndpointReference>
(22)      </d:Bye>
(23)    </s:Body>
(24) </s:Envelope>
```

Lines (06-08) indicate this is a Bye, and like the Hello in Table 6, the distinguished URI in Line (12) indicates it is a multicast Bye sent over the multicast channels listed in Section 2.4 Protocol Assignments. The sequence information in Line (13) indicates this message is to be ordered after the Hello in Table 6 because the Bye has a larger message number than the Hello within the same instance identifier. Note that the Body (Lines 16-22) is an abbreviated form of the corresponding information in the Hello; when a Target Service leaves a network, it is sufficient to send the stable identifier to indicate the Target Service is no longer available.

5. Probe and Probe Match

Messages described in this section MUST be implemented by a Target Service, MUST be implemented by a Discovery Proxy (for itself, not for other Target Services), and MAY be implemented by a Client.

5.1 Matching Types and Scopes

A Probe includes zero, one, or two constraints on matching Target Services: a Type and/or a Scope. This section defines the matching rules between one of these in a Probe and a member of the corresponding set associated with a Target Service.

A Type T1 in a Probe matches Type T2 if the QNames match. Specifically, T1 matches T2 if all of the following are true:

- The namespace (see [Namespaces in XML 1.1]) of T1 and T2 are the same.
- The local name of T1 and T2 are the same.

(The namespace prefix of T1 and T2 is relevant only to the extent that it identifies the namespace.)

A Scope S1 in a Probe matches Scope S2 per the rule indicated within the Probe. The matching rules defined below are expected to be widely useful.

http://schemas.xmlsoap.org/ws/2004/02/discovery/rfc2396

Using a case-insensitive comparison,

- The `scheme` (see [RFC 2396]) of S1 and S2 is the same and
- The `hostname` of S1 and S2 is the same and
- The `path_segments` of S1 is a `segment`-wise (not string) prefix of the `path_segments` of S2, ignoring any `param` components of `segment` components and
- Neither S1 nor S2 contain the "." `segment` or the ".." `segment`.

All other components (e.g., `query` and `fragment`) are explicitly excluded from comparison. S1 and S2 MUST be canonicalized (e.g., unescaping escaped characters) before using this matching rule. Note: this is NOT whether the string representation of S1 is a prefix of the string representation of S2.

http://schemas.xmlsoap.org/ws/2004/02/discovery/uuid
: The scheme of S1 and S2 is the same and each of the unsigned integer fields (see [UUID]) in S1 is equal to the corresponding field in S2, or equivalently, the 128 bits of the in-memory representation of S1 and S2 are the same 128 bit unsigned integer.

http://schemas.xmlsoap.org/ws/2004/02/discovery/ldap
: The scheme of S1 and S2 is the same and the `hostport` (see [RFC 2255]) of S1 and S2 is the same and the `RDNSequence` (see [RFC 2253]) of the `dn` of S1 is a prefix of the `RDNSequence` of the `dn` of S2.

http://schemas.xmlsoap.org/ws/2004/02/discovery/strcmp0
: Using a case-sensitive comparison, the string representation of S1 and S2 is the same.

Other matching rules MAY be used, but if a matching rule is not recognized by a receiver of the Probe, S1 does not match S2.

5.2 Probe

If a Client has not discovered any Discovery Proxies, it MAY send a Probe to find Target Services of a given Type and/or in a given Scope. Such a Client MAY also send a Probe to find Target Services regardless of their Types or Scopes.

A Probe is a one-way message. If a Client knows only the type of scope of Target Services, or is interested in discovering all Target Services, the Probe is sent to the IPv4 and/or IPv6 multicast addresses listed in Section 2.4 Protocol Assignments.

However, if a Client knows the network address of a Target Service, the Probe MAY be sent directly to that network address; note that in this case, other semantics of the Probe or Probe Match are unchanged, e.g., the Probe MAY include a type, it is sent to the same logical address (SOAP To header block), the receiver MAY not respond, etc.

Because a Client does not know in advance how many Target Services (if any) will respond, the Client MAY adopt either of the following behaviors:

- Wait for a sufficient number of responses.
- Repeat the Probe several times until the Client is convinced that no further responses will be received. The Client MUST use the same value for the MessageID SOAP header block (see [WS-Addressing]) in all copies of the Probe.

The normative outline for Probe is:

```
<s:Envelope ...>
  <s:Header ...>
    <a:Action ...>
        http://schemas.xmlsoap.org/ws/2004/02/discovery/Probe
    </a:Action>
    <a:MessageID ...>xs:anyURI</a:MessageID>
   [<a:ReplyTo ...>
      <a:Address ...>xs:anyURI</a:Address>
      ...
    </a:ReplyTo>]?
    <a:To ...>http://schemas.xmlsoap.org/ws/2004/02/discovery</a:To>
    ...
  </s:Header>
  <s:Body ...>
    <d:Probe ...>
     [<d:Types>list of xs:QName</d:Types>]?
     [<d:Scope [MatchBy='xs:anyURI']?>xs:anyURI</d:Scope>]?
     ...
    </d:Probe>
  </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/a:Action

If a SOAP Action URI is used in the binding for SOAP, the value indicated herein MUST be used for that URI.

/s:Envelope/s:Header/a:ReplyTo

If included, MUST be of type a:EndpointReferenceType (see [WS-Addressing]). If omitted, implied value of the **[reply endpoint]** property (see [WS-Addressing]) of the Probe is the source address and port number of the Probe transport header.

/s:Envelope/s:Header/a:ReplyTo/a:Address

If the value is 'http://schemas.xmlsoap.org/ws/2003/03/addressing/role/anonymous', the address of the **[reply endpoint]** property of the Probe is the source address and port number of the Probe transport header.

/s:Envelope/s:Header/a:To

(Note that a Client may only send a Probe if it has not discovered any Discovery Proxies.)

/s:Envelope/s:Body/d:Probe/d:Types

If omitted, implied value is any Type. If a Target Service has no Type that matches any of these values (see Section 5.1 Matching Types and Scopes), the Target Service MUST NOT respond.

/s:Envelope/s:Body/d:Probe/d:Scope
If included, MUST be an absolute URI. If omitted, implied value is any Scope. If a Target Service has no Scope that matches this value, the Target Service MUST NOT respond.

/s:Envelope/s:Body/d:Probe/d:Scope/@MatchBy
If omitted, implied value is 'http://schemas.xmlsoap.org/ws/2004/02/discovery/rfc2396'.

To Probe for all Target Services, a Client MAY omit both /s:Envelope/s:Body/d:Probe/d:Types and ./d:Scope.

5.3 Probe Match

If a Target Service matches a Probe, the Target Service MUST respond with a Probe Match message.

A Target Service MUST wait for a timer after receiving a Probe before sending a Probe Match as described in Section 2.4 Protocol Assignments.

A Probe Match MUST be unicast to the **[reply endpoint]** property (see [WS-Addressing]) of the Probe.

The normative outline for Probe Match is:

```
<s:Envelope ...>
  <s:Header ...>
    <a:Action ...>
        http://schemas.xmlsoap.org/ws/2004/02/discovery/ProbeMatch
    </a:Action>
    <a:MessageID ...>xs:anyURI</a:MessageID>
    <a:RelatesTo ...>xs:anyURI</a:RelatesTo>
    <a:To ...>xs:anyURI</a:To>
    ...
  </s:Header>
  <s:Body ...>
    <d:ProbeMatch ...>
      <a:EndpointReference ...>
        <a:Address ...>xs:anyURI</a:Address>
       [<a:ReferenceProperties ...>...</a:ReferenceProperties>]?
        ...
       [<p:Policy>policy expression</p:Policy>]?
      </a:EndpointReference>
     [<d:Types ...>list of xs:QName</d:Types>]?
```

```
     [<d:Scopes ...>list of xs:anyURI</d:Scopes>]?
      <d:MetadataVersion ...>xs:nonNegativeInteger</d:MetadataVersion>
      ...
    </d:ProbeMatch>
  </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/a:Action
: If a SOAP Action URI is used in the binding for SOAP, the value indicated herein MUST be used for that URI.

/s:Envelope/s:Header/a:RelatesTo
: MUST be the value of /s:Envelope/s:Header/a:MessageID of the Probe.

/s:Envelope/s:Header/a:To
: If value of the **[reply endpoint]** property (see [WS-Addressing]) of the Probe is the source address and port number of the Probe transport header, e.g., when the a:ReplyTo header block was omitted from the corresponding Probe, MUST be 'http://schemas.xmlsoap.org/ws/2003/03/addressing/role/anonymous'.

/s:Envelope/s:Body/d:ProbeMatch/*
: As constrained for Hello (see Section 4.1 Hello).

6. Resolve and Resolve Match

Messages described in this section MUST be implemented by a Target Service, MUST be implemented by a Discovery Proxy (for itself, not for other Target Services), and MAY be implemented by a Client.

6.1 Resolve

If a Client has an Endpoint Reference [WS-Addressing] for a Target Service, and does not have enough metadata to bootstrap communication with the Target Service, and has not discovered any Discovery Proxies, the Client MAY send a Resolve message to the Endpoint Reference for the Target Service.

A Resolve is a one-way message sent using the IPv4 and/or IPv6 multicast addresses listed in Section 2.4 Protocol Assignments.

The normative outline for Resolve is:

```
<s:Envelope ...>
  <s:Header ...>
    <a:Action ...>
        http://schemas.xmlsoap.org/ws/2004/02/discovery/Resolve
    </a:Action>
    <a:MessageID ...>xs:anyURI</a:MessageID>
   [<a:ReplyTo ...>
      <a:Address ...>xs:anyURI</a:Address>
      ...
```

```
    </a:ReplyTo>]?
    <a:To ...>xs:anyURI</a:To>
    ...
  </s:Header>
  <s:Body ... />
</s:Envelope>
```

The following describes normative constraints on the outline above:

/s:Envelope/s:Header/a:Action
: If a SOAP Action URI is used in the binding for SOAP, the value indicated herein MUST be used for that URI.

/s:Envelope/s:Header/a:ReplyTo
: If included, MUST be of type a:EndpointReferenceType (see [WS-Addressing]). If omitted, implied value of the **[reply endpoint]** property (see [WS-Addressing]) of the Resolve is the source address and port number of the Resolve transport header.

/s:Envelope/s:Header/a:ReplyTo/a:Address
: If the value is 'http://schemas.xmlsoap.org/ws/2003/03/addressing/role/anonymous', the address of the **[reply endpoint]** property of the Resolve is the source address and port number of the Resolve transport header.

/s:Envelope/s:Header/a:To
: Per WS-Addressing [WS-Addressing], MUST be a:EndpointReference/a:Address for the Target Service. MUST NOT be 'http://schemas.xmlsoap.org/ws/2004/02/discovery'.

Per WS-Addressing [WS-Addressing], messages sent to an Endpoint Reference MUST include children of a:EndpointReference/a:ReferenceProperties as SOAP header blocks. If the identity of the Endpoint Reference for a Target Service does not match /s:Envelope/s:Header/a:To and these header blocks, the Target Service MUST NOT respond.

6.2 Resolve Match

If a Target Service matches a Resolve, the Target Service MUST respond with a Resolve Match message.

A Target Service MUST send Resolve Match without waiting for a timer.

A Resolve Match MUST be unicast to the **[reply endpoint]** property (see [WS-Addressing]) of the Resolve.

The normative outline for Resolve Match is:

```
<s:Envelope ...>
  <s:Header ...>
    <a:Action ...>
        http://schemas.xmlsoap.org/ws/2004/02/discovery/ResolveMatch
    </a:Action>
    <a:MessageID ...>xs:anyURI</a:MessageID>
```

```
    <a:RelatesTo ...>xs:anyURI</a:RelatesTo>
    <a:To ...>xs:anyURI</a:To>
    ...
  </s:Header>
  <s:Body ...>
    <d:ResolveMatch ...>
      <a:EndpointReference ...>
        <a:Address ...>xs:anyURI</a:Address>
       [<a:ReferenceProperties ...>...</a:ReferenceProperties>]?
        ...
       [<p:Policy>policy expression</p:Policy>]?
      </a:EndpointReference>
     [<d:Types ...>list of xs:QName</d:Types>]?
     [<d:Scopes ...>list of xs:anyURI</d:Scopes>]?
      <d:MetadataVersion ...>xs:nonNegativeInteger</d:MetadataVersion>
      ...
    </d:ResolveMatch>
  </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/a:Action
: If a SOAP Action URI is used in the binding for SOAP, the value indicated herein MUST be used for that URI.

/s:Envelope/s:Header/a:RelatesTo
: MUST be the value of /s:Envelope/s:Header/a:MessageID of the Resolve.

/s:Envelope/s:Header/a:To
: If value of the **[reply endpoint]** property (see [WS-Addressing]) of the Resolve is the source address and port number of the Resolve transport header, MUST be 'http://schemas.xmlsoap.org/ws/2003/03/addressing/role/anonymous'.

/s:Envelope/s:Body/d:ResolveMatch/*
: As constrained for Hello (see Section 4.1 Hello).

7. Security Model

This specification does not require that endpoints and the discovery process be secure. However, this specification RECOMMENDS that security be used to mitigate many types of attacks. The security model for discovery builds on the model described in WSS-SOAP Message Security (see [WSS-SOAPSecurity]), WS-Trust (see [WS-Trust]), and WS-Federation (see [WS-Federation]). This section outlines how these specifications are used to secure the discovery process.

The discovery process begins with the Hello message when a Target Service announces itself. It is RECOMMENDED that the Target Service secure its

announcement to prevent an antagonist from pretending to be a Target Service (as well as other potential threats). However, Target Service MAY use a variety of security mechanisms, some of which may not be available to all recipients of the Hello. Consequently, it is assumed that appropriate recipients (meaning it is not required that all recipients) be able to verify the integrity and authenticate the Hello. This implies that at least one of the following to be true:

- The Target Service uses a public key whose certification can be evaluated.
- The appropriate recipients have been previously given (potentially using out-of-scope or out-of-band mechanisms) the required information in order to validate the Hello.
- The appropriate recipients have the ability to obtain and validate the Hello using out-of-scope or out-of-band mechanisms.

If the announcement needs to be confidential, then the body of the message is encrypted using the mechanisms described in WSS-SOAP Message Security. If the source of the message needs to be secured, then the contents of the appropriate headers (e.g., From, ReplyTo, etc.), not the header itself, are encrypted.

It is equally important, and also RECOMMENDED, that the Bye message be secured to prevent unauthorized Bye messages. The Bye can be secured using any token that is understood by the appropriate recipients.

If a Hello or Bye is secured, and the mechanism or token is not understood, recipients SHOULD ignore the message due to the security risks associated with processing it. If the authentication mechanisms are not pre-defined (meaning the exact key was provided out-of-band), then steps should be taken to ensure that the Target Service has the right to speak for the specified endpoint.

As a general interoperability point, it is RECOMMENDED that public-key security tokens and XML Signature recommended algorithms be used to simplify and increase interoperability of the verification process.

While the Probe and Resolve are NOT REQUIRED to be secured, doing so is RECOMMENDED to prevent various attacks. If any expose restricted data then they SHOULD be secured to protect the data. The use of WSS-SOAP Message Security to secure messages is strongly RECOMMENDED. Recipients of Probe and Resolve messages should ensure that the requestor has access to the following:

- The Target Service to which the request was sent.
- The Types listed in d:Types.
- The Scopes identified in d:Scopes.
- The right to speak for the service to which the response is sent (e.g., ReplyTo header block, see [WS-Addressing]).

The last point, "speaks for," refers to the validation of the claims that the requestor makes. For example, if the message claims to be from (or wants replies sent to) example.com, then it must prove that it represents example.com. This may be done, for example, using a certification from a trusted source indicating the rights of example.com. This is similar to what happens in HTTPS today where the certificate of the site is validated against the URL entered.

Similarly, the Probe Match and Resolve Match messages are NOT REQUIRED to be secured, but doing so is also RECOMMENDED. Recipients should ensure that the sender has the right to speak for the specified endpoints identified in the response.

In some cases, additional capability tokens are specified in Probe (or Resolve) and Probe Match (or Resolve Match) messages to indicate authorization to find and report different services. For example, one token may be provided to prove identity, but another token may be needed to indicate that the specified identity has the right to issue Probes against the specified target.

If a Target Service determines it is at risk by information disclosure threats, it MAY choose to not respond to specific Probe messages, even if the Target Service matches the Probe. Moreover, it MAY choose to limit the Types and/or Scopes it lists in a Hello, Probe Match, and/or Resolve Match response.

Access control MAY be applied to any request. That is, if a request is made, which the recipient does not take to be properly authorized, it MAY fail the request. Additionally, if a service believes that it is under attack or for any reasons believes it would be a risk to reply or issue a fault, it MAY omit (drop) its response.

8. Security Considerations

Message discovery, both announcements and searches, are subject to a wide variety of attacks and therefore it is strongly RECOMMENDED that the communication be secured using the mechanisms described in WSS-SOAP Message Security (see [WSS-SOAPSecurity]).

In order to properly secure messages, the body and all relevant headers need to be included in the signature. Specifically, the WS-Addressing (see [WS-Addressing]) headers and any headers identified in endpoint references need to be signed with the body (and other key headers) in order to "bind" the two together and prevent alteration/replay.

In the event that a service communicates frequently with a discovery service, it is RECOMMENDED that a security context be established using the mechanisms described in WS-Trust (see [WS-Trust]) and WS-SecureConversation (see [WS-SecureConversation]) allowing for potentially more efficient means of authentication.

To maximize the strength of shared keys/secrets, it is RECOMMENDED that the keys be changed frequently. This "re-keying" can be effected in a number of ways. The following list outlines four common techniques:

- Attaching a nonce to each message and using it in a derived key function with the shared secret
- Using a derived key sequence and switch "generations"
- Closing and re-establishing a security context (not possible for delegated keys)
- Exchanging new secrets between the parties (not possible for delegated keys)

It should be noted that the mechanisms listed above are independent of the SCT (see [WS-Trust, WS-SecureConversation]) and secret returned when a context (or any shared secret) is created. That is, the keys used to secure the channel may be independent of the key used to prove the right to access a service.

The security context MAY be re-established using the mechanisms described in WS-Trust and WS-SecureConversation. Similarly, secrets can be exchanged using the mechanisms described in WS-Trust. Note, however, that the current shared secret SHOULD NOT be used to encrypt the new shared secret. Derived keys, the preferred solution from this list, can be specified using the mechanisms described in WS-SecureConversation.

The following list summarizes common classes of attacks that apply to this protocol and identifies the mechanism to prevent/mitigate the attacks:

- **Message alteration** – Alteration is prevented by including signatures of the message information using WSS-SOAP Message Security.
- **Message disclosure** – Confidentiality is preserved by encrypting sensitive data using WSS-SOAP Message Security.
- **Key integrity** – Key integrity is maintained by using the strongest algorithms possible (by comparing secured policies – see [WS-SecurityPolicy]).
- **Authentication** – Authentication is established using the mechanisms described in WSS-SOAP Message Security and WS-Trust. Each message is authenticated using the mechanisms described in WSS-SOAP Message Security.
- **Accountability** – Accountability is a function of the type of and string of the key and algorithms being used. In many cases, a strong symmetric key provides sufficient accountability. However, in some environments, strong PKI signatures are required.
- **Availability** – Many services are subject to a variety of availability attacks. Replay is a common attack and it is RECOMMENDED that this be addressed as described in the next bullet. Other attacks, such as network-level denial of service attacks are harder to avoid and are beyond the scope of this specification. That said, care should be taken to ensure that minimal processing be performed prior to any authenticating sequences.
- **Replay** – Messages may be replayed for a variety of reasons. To detect and eliminate this attack, mechanisms should be used to identify replayed messages such as the timestamp/nonce outlined in WSS-SOAP Message Security. Alternatively, and optionally, other technologies, such as sequencing, can also be used to prevent replay of application messages.

9. Acknowledgements

This specification has been developed as a result of joint work with many individuals and teams, including: Don Box (Microsoft), Mike Fenelon (Microsoft), Omri Gazitt (Microsoft), Richard Hasha (Microsoft), Erin Honeycutt (Microsoft), Christian Huitema (Microsoft), Chris Kaler (Microsoft), Thomas Kuehnel (Microsoft), Umesh Madan (Microsoft), Vipul Modi (Microsoft), Jeff Parham (Microsoft), Yaniv Pessach (Microsoft), Stefan Pharies (Microsoft), Dale Sather (Microsoft), Matt Tavis (Microsoft), and Eugene Yarmosh (Intel).

10. References

[IANA]
Port Numbers, February 2004.

[Namespaces in XML 1.1]
T. Bray, et al, "Namespaces in XML 1.1," February 2004.

[RFC 792]
J. Postel, "Internet Control Message Protocol," RFC 792, ISI, September 1981.

[RFC 2119]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997.

[RFC 2253]

M. Wahl, et al, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names," RFC 2253, December 1997.

[RFC 2255]
T. Howes, et al, "The LDAP URL Format," RFC 2255, December 1997.

[RFC 2396]
T. Berners-Lee, et al, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, August 1998.

[SOAP 1.1]
D. Box, et al, "Simple Object Access Protocol (SOAP) 1.1," May 2000.

[SOAP 1.2]
M. Gudgin, et al, "SOAP Version 1.2 Part 1: Messaging Framework," June 2003.

[SOAP/UDP]
M. Gudgin, et al, "SOAP over UDP," to be published.

[UUID]
P. Leach, et al, "UUIDs and GUIDs: draft-leach-uuids-guids-01.txt," February 1998.

[WS-Addressing]
A. Bosworth, et al, "Web Services Addressing (WS-Addressing)," March 2003.

[WS-Federation]
S. Bajaj, et al, "Web Services Federation Language (WS-Federation)," July 2003.

[WS-Policy]
D. Box, et al, "Web Services Policy Framework (WS-Policy)," May 2003.

[WS-SecureConversation]
G. Della-Libera, et al, "Web Services Secure Conversation Language (WS-SecureConversation)," December 2002.

[WS-SecurityPolicy]
G. Della-Libera, et al, "Web Services Security Policy Language (WS-SecurityPolicy)," December 2002.

[WS-Trust]
G. Della-Libera, et al, "Web Services Trust Language (WS-Trust)," December 2002.

[WSDL 1.1]
E. Christensen, et al, "Web Services Description Language (WSDL) 1.1," March 2001.

[WSS-SOAPSecurity]
A. Nadalin, et al, "Web Services Security: SOAP Message Security," August 2003.

[XML Infoset]
J. Cowan, et al, "XML Information Set (Second Edition)," February 2004.

[XML Schema, Part 1]
H. Thompson, et al, "XML Schema Part 1: Structures," May 2001.

[XML Schema, Part 2]
P. Biron, et al, "XML Schema Part 2: Datatypes," May 2001.

[XPath 1.0]
J. Clark, et al, "XML Path Language (XPath) Version 1.0," November 1999.

Appendix I – Application Sequencing

The Application Sequencing header block allows a receiver to order messages that contain this header block though they might have been received out of order. It is used by this specification to allow ordering Hello and Bye messages from a Target Service; it is also expected that this header block will be useful in other applications.

The normative outline for the application sequence header block is:

```
<s:Envelope ...>
  <s:Header ...>
    <d:AppSequence InstanceId='xs:nonNegativeInteger'
                  [SequenceId='xs:anyURI']?
                   MessageNumber='xs:nonNegativeInteger'
                   ... />
  </s:Header>
  <s:Body ...> ... </s:Body>
</s:Envelope>
```

The following describes normative constraints on the outline listed above:

/s:Envelope/s:Header/d:AppSequence/@InstanceId
: MUST be incremented by >= 1 each time the service has gone down, lost state, and came back up again. SHOULD NOT be incremented otherwise. Means to set this value include, but are not limited to:
  - A counter that is incremented on each 'cold' boot
  - The boot time of the service, expressed as seconds elapsed since midnight January 1, 1970

/s:Envelope/s:Header/d:AppSequence/@SequenceId
: Identifies a sequence within the context of an instance identifier. If omitted, implied value is the null sequence. MUST be unique within ./@InstanceId.

/s:Envelope/s:Header/d:AppSequence/@MessageNumber
: Identifies a message within the context of a sequence identifier and an instance identifier. MUST be incremented by >= 1 for each message sent. Transport-level retransmission MUST preserve this value.

Other components of the outline above are not further constrained by this specification.

Appendix II – Provisional Policy Assertions

A Target Service MAY be reachable over various transports, using various network addresses. To indicate this, appropriate policy assertions (see [WS-Policy]) SHOULD be included in a Hello, Probe Match, and Resolve Match if the unique, stable identifier for a Target Service is not resolvable to a transport and network address.

It is expected that other specifications will define assertions to indicate both: (a) a well-defined algorithm for turning an encoded byte array, which represents the XML Infoset of a SOAP message, into transport packets, and (b) a transport address to which the transport packets are sent.

However, to serve as the basis for interoperability, one such assertion is provisionally defined herein to indicate SOAP over HTTP/1.1 as defined in Section 3 of WSDL 1.1 [WSDL 1.1] (generalized in the obvious way to apply to either SOAP 1.1 or SOAP 1.2). The normative outline for this assertion is:

```
<d:SoapHttpRequestReplyAddress ... >
  xs:anyURI
</d:SoapHttpRequestReplyAddress>
```

The following describes normative constraints on the outline above:

/d:SoapHttpRequestReplyAddress
: MUST be a request URI whose scheme is "http" and MUST be resolvable to a network address.

This assertion MAY be repeated within a WS-Policy `p:OneOrMore` operator to indicate the Target Service is reachable using this transport over multiple addresses. This assertion MUST NOT be repeated within a policy alternative (typically through a WS-Policy `p:All` operator).

Appendix III – XML Schema

A normative copy of the XML Schema [XML Schema Part 1, Part 2] and WSDL [WSDL 1.1] descriptions for this specification may be retrieved by resolving the XML namespace URI for this specification (listed in Section 2.3 XML Namespaces). A non-normative copy of the XML Schema description is listed below for convenience.

Normative text within this specification takes precedence over the normative outlines, which take precedence over the XML Schema and WSDL descriptions.

```
<xs:schema
    targetNamespace='http://schemas.xmlsoap.org/ws/2004/02/discovery'
    xmlns:tns='http://schemas.xmlsoap.org/ws/2004/02/discovery'
    xmlns:wsa='http://schemas.xmlsoap.org/ws/2003/03/addressing'
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    elementFormDefault='qualified'
    blockDefault='#all' >

  <xs:import
      namespace='http://schemas.xmlsoap.org/ws/2003/03/addressing'
      schemaLocation='http://schemas.xmlsoap.org/ws/2003/03/addressing'
/>

  <!-- ////////////////////// WS-Discovery /////////////////////// -->

  <!-- Messages -->
```

```
<xs:element name='Hello' type='tns:DiscoveryReferenceType' />

<xs:complexType name='DiscoveryReferenceType' >
  <xs:sequence>
    <xs:element ref='wsa:EndpointReference' />
    <xs:element ref='tns:Types' minOccurs='0' />
    <xs:element ref='tns:Scopes' minOccurs='0' />
    <xs:element ref='tns:MetadataVersion' />
    <xs:any namespace='##other'
            processContents='lax'
            minOccurs='0'
            maxOccurs='unbounded' />
  </xs:sequence>
  <xs:anyAttribute namespace='##other' processContents='lax' />
</xs:complexType>

<xs:element name='Bye' >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='wsa:EndpointReference' />
      <xs:any namespace='##any'
              processContents='lax'
              minOccurs='0'
              maxOccurs='unbounded' />
      <!-- ##any allows simplifying an implementation to
           send the same children in Bye as in Hello -->
    </xs:sequence>
    <xs:anyAttribute namespace='##other' processContents='lax' />
  </xs:complexType>
</xs:element>

<xs:element name='Probe' >
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='tns:Types' minOccurs='0' />
      <xs:element ref='tns:Scope' minOccurs='0' />
```

```
      <xs:any namespace='##other'
              processContents='lax'
              minOccurs='0'
              maxOccurs='unbounded' />
    </xs:sequence>
    <xs:anyAttribute namespace='##other' processContents='lax' />
  </xs:complexType>
</xs:element>

<xs:element name='ProbeMatch' type='tns:DiscoveryReferenceType' />

<!-- No body defined for Resolve -->

<xs:element name='ResolveMatch' type='tns:DiscoveryReferenceType' />

<!-- Shared global elements -->

<xs:element name='Types' >
  <xs:simpleType>
    <xs:list itemType='xs:QName' />
  </xs:simpleType>
</xs:element>

<xs:element name='Scope' >
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='xs:anyURI' >
        <xs:attribute name='MatchBy' type='xs:anyURI' />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name='Scopes' >
  <xs:simpleType>
    <xs:list itemType='xs:anyURI' />
```

```
    </xs:simpleType>
  </xs:element>

  <xs:element name='MetadataVersion' type='xs:nonNegativeInteger' />

  <!-- ////////////////////// General Headers /////////////////////// -->

  <xs:element name='AppSequence' >
   <xs:complexType>
    <xs:complexContent>
     <xs:restriction base='xs:anyType' >
      <xs:attribute name='InstanceId'
                    type='xs:positiveInteger'
                    use='required' />
      <xs:attribute name='SessionId' type='xs:anyURI' />
      <xs:attribute name='MessageNumber'
                    type='xs:positiveInteger'
                    use='required' />
     </xs:restriction>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>

  <!-- ////////////////////// Policy Assertions /////////////////////// -->

  <xs:element name='SoapHttpRequestReplyAddress' >
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:anyURI' >
          <xs:anyAttribute namespace='##other' processContents='lax' />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Appendix IV – WSDL

A non-normative copy of the WSDL [WSDL 1.1] description for this specification is listed below for convenience.

```
<wsdl:definitions
    targetNamespace='http://schemas.xmlsoap.org/ws/2004/02/discovery'
    xmlns:tns='http://schemas.xmlsoap.org/ws/2004/02/discovery'
    xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
    xmlns:xs='http://www.w3.org/2001/XMLSchema' >

  <wsdl:types>
    <xs:schema>
      <xs:import
          namespace='http://schemas.xmlsoap.org/ws/2004/02/discovery'
          schemaLocation='discovery.xsd' />
    </xs:schema>
  </wsdl:types>

  <wsdl:message name='HelloMsg' >
    <wsdl:part name='body' element='tns:Hello' />
  </wsdl:message>

  <wsdl:message name='ByeMsg' >
    <wsdl:part name='body' element='tns:Bye' />
  </wsdl:message>

  <wsdl:message name='ProbeMsg' >
    <wsdl:part name='body' element='tns:Probe' />
  </wsdl:message>

  <wsdl:message name='ProbeMatchMsg' >
    <wsdl:part name='body' element='tns:ProbeMatch' />
  </wsdl:message>

  <wsdl:message name='ResolveMsg' />

  <wsdl:message name='ResolveMatchMsg' >
```

```
  <wsdl:part name='body' element='tns:ResolveMatch' />
</wsdl:message>

<wsdl:portType name='Discovery' >
  <wsdl:operation name='HelloOp' >
    <wsdl:output message='tns:HelloMsg' />
  </wsdl:operation>
  <wsdl:operation name='ByeOp' >
    <wsdl:output message='tns:ByeMsg' />
  </wsdl:operation>
  <wsdl:operation name='ProbeOp' >
    <wsdl:input message='tns:ProbeMsg' />
  </wsdl:operation>
  <wsdl:operation name='ProbeMatchOp' >
    <wsdl:output message='tns:ProbeMatchMsg' />
  </wsdl:operation>
</wsdl:portType>

<wsdl:portType name='TargetService' >
  <wsdl:operation name='ResolveOp' >
    <wsdl:input message='tns:ResolveMsg' />
  </wsdl:operation>
  <wsdl:operation name='ResolveMatchOp' >
    <wsdl:output message='tns:ResolveMatchMsg' />
  </wsdl:operation>
</wsdl:portType>

<!-- If this portType is included in EndpointReference/Types, it
     indicates the Target Service is a Discovery Proxy.  Because
     this specification does not define proxy-specific operations,
     no operations are defined within this portType; Discovery
     Proxies implement other types that define proxy-specific
     operations.  Discovery Proxies also implement
     TargetService. -->
<wsdl:portType name='DiscoveryProxy' />
```

```
</wsdl:definitions>
```