Web Services Dynamic Discovery (WS-Discovery) Version 1.1

Committee Draft 01

27 January 2009

Specification URIs:

This Version:

http://docs.oasis-open.org/ws-dd/discovery/1.1/cd-01/wsdd-discovery-1.1-spec-cd-01.html
http://docs.oasis-open.org/ws-dd/discovery/1.1/cd-01/wsdd-discovery-1.1-spec-cd-01.docx
(Authoritative Format)
http://docs.oasis-open.org/ws-dd/discovery/1.1/cd-01/wsdd-discovery-1.1-spec-cd-01.pdf

Previous Version:

N/A

Latest Version:

http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html
http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.docx
http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.pdf

Technical Committee:

OASIS Web Services Discovery and Web Services Devices Profile (WS-DD) TC

Chair(s):

Toby Nixon, Microsoft Corporation
Alain Regnier, Ricoh Company Limited

Editor(s):

Vipul Modi, Microsoft Corporation
Devon Kemp, Canon Inc.

Declared XML Namespace(s):

http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09

Abstract:

This specification defines a discovery protocol to locate services. In an ad hoc mode of operation, 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 and to extend the reach of the protocol, this protocol defines a managed mode of operation and a 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.

Status:

This document was last revised or approved by the WS-DD TC on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at http://www.oasis-open.org/committees/ws-dd/.

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

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/ws-dd/.

Notices

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

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

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

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

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

Table of Contents

1 Introduction

This specification defines a discovery protocol to locate services. The primary mode of discovery is a client searching for one or more target services. The protocol defines two modes of operation, an ad hoc mode and a managed mode. In an ad hoc mode, 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 in an ad hoc network, 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 and to extend the reach of the protocol beyond the range of an ad hoc network, this specification defines a managed mode of operation and a multicast suppression behavior if a discovery proxy is available on the network. In managed mode, target services sent unicast announcement messages to a discovery proxy and clients send unicast probe and resolve messages to a discovery proxy. To reduce multicast traffic, when a discovery proxy detects a probe or resolution request sent multicast on an ad hoc network, it sends an announcement for itself. By listening for these announcements, clients detect discovery proxies and switch to a managed mode of operation and send unicast probe and resolve messages directly to a discovery proxy. However, if a discovery proxy is unresponsive, clients revert to an ad hoc mode of operation.

To support networks with explicit network management services like DHCP, DNS, domain controllers, directories, etc., 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 client and target services to operate in a managed mode and 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 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.

1.2 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 and allow discovery of services in managed networks where such network 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.

1.3 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.

1.4 Example

Table 1 lists an example Probe message sent multicast by a Client searching for a printer in an ad hoc mode.

Table 1: Example Probe sent multicast in an ad hoc mode.

```
(01) <s:Envelope
(02)     xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(03)     xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09"
(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://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/Probe
(09)     </a:Action>
(10)     <a:MessageID>
(11)       urn:uuid:0a6dc791-2be6-4991-9af1-454778a1917a
(12)     </a:MessageID>
(13)     <a:To>urn:docs-oasis-open-org:ws-dd:discovery:2008:09</a:To>
(14)   </s:Header>
(15)   <s:Body>
(16)     <d:Probe>
(17)       <d:Types>i:PrintBasic</d:Types>
(18)       <d:Scopes
(19)    MatchBy="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/ldap" >
(20)         ldap:///ou=engineering,o=examplecom,c=us
(21)       </d:Scopes>
(22)     </d:Probe>
(23)   </s:Body>
(24) </s:Envelope>
(25)
```

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 [RFC 2141].

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

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 ProbeMatch sent in response to the ad hoc Probe in Table 1.

```
(01) <s:Envelope
(02)   xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(03)   xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09"
(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://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/ProbeMatches
```

```
(09)     </a:Action>
(10)     <a:MessageID>
(11)       urn:uuid:e32e6863-ea5e-4ee4-997e-69539d1ff2cc
(12)     </a:MessageID>
(13)     <a:RelatesTo>
(14)       urn:uuid:0a6dc791-2be6-4991-9af1-454778a1917a
(15)     </a:RelatesTo>
(16)     <a:To>
(17)     http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
(18)     </a:To>
(19)     <d:AppSequence InstanceId="1077004800" MessageNumber="2" />
(20)   </s:Header>
(21)   <s:Body>
(22)     <d:ProbeMatches>
(23)       <d:ProbeMatch>
(24)         <a:EndpointReference>
(25)           <a:Address>
(26)             urn:uuid:98190dc2-0890-4ef8-ac9a-5940995e6119
(27)           </a:Address>
(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)           http://itdept/imaging/deployment/2004-12-04
(34)         </d:Scopes>
(35)         <d:XAddrs>http://prn-example/PRN42/b42-1668-a</d:XAddrs>
(36)         <d:MetadataVersion>75965</d:MetadataVersion>
(37)       </d:ProbeMatch>
(38)     </d:ProbeMatches>
(39)   </s:Body>
(40) </s:Envelope>
(41)
```

Lines (07-09) in Table 2 indicate this message is a Probe Match, and Lines (13-15) 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 (16-18) indicate that the response was sent to the source IP address and port of the transport header of the Probe. Line (19) contains an instance identifier as well as a message number; this information allows the receiver to reorder discovery messages received from a Target Service.

Lines (23-37) describe a single Target Service.

Lines (24-28) contain the stable, unique identifier for the Target Service that is constant across network interfaces, transport addresses, and IPv4/v6. In this case, the value is a UUID based URN [RFC 4122] scheme URI, but it may be a transport URI (like the one in Line 35) if it meets stability and uniqueness requirements.

Line (29) lists the Types (see, e.g., [WSDL 1.1]) implemented by the Target Service, in this example, a basic print type that matched the Probe as well as an advanced print type.

Lines (30-34) list three administrative Scopes, one that matched the Probe (Line 31), one that is specific to a particular physical location (Line 32), and one that includes data useful when switching over to new infrastructure (Line 33). As in this case, the Scopes may be a heterogeneous collection of deployment-related information.

Line (35) indicates the transport addresses where the Target Service may be reached; in this case, a single HTTP transport address.

Line (36) 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 29-34).

1.5 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC 2119].

1.5.1 Notational Conventions

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 literal 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 points of extensibility. 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.

Elsewhere in this specification, the characters "[" and "]" are used to call out references and property names. This specification uses the **[action]** and Fault properties [WS-Addressing] to define faults.

1.5.2 Terms and Definitions

Defined below are the basic definitions for the terms used in this specifications.

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 of Target Services by Clients.

Hello: A message sent by a Target Service when it joins a network; this message contains key information for the Target Service. A Hello message is also sent by a Discovery Proxy to reduce multicast traffic on an ad hoc network; this message contains key information about the Discovery Proxy.

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 allows Target Services to be organized into logical groups.

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

Ad hoc Mode: An operational mode of discovery in which the Hello, Bye, Probe and Resolve messages are sent multicast.

Managed Mode: An operational mode of discovery in which the Hello, Bye, Probe and Resolve message are sent unicast to a Discovery Proxy.

Ad hoc Network: A network in which discovery is performed in an ad hoc mode.

Managed Network: A network in which discovery is performed in a managed mode.

1.6 XML Namespaces

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

```
http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09
```

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: Prefix 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/	[SOAP 1.1]
s12	http://www.w3.org/2003/05/soap-envelope	[SOAP 1.2]
a	http://schemas.xmlsoap.org/ws/2004/08/addressing	[WS-Addressing]
d	http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09	This specification
ds	http://www.w3.org/2000/09/xmldsig#	[XML Sig]
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd	[WS-Security]
xs	http://www.w3.org/2001/XMLSchema	[XML Schema Part 1, 2]

1.7 XSD and WSDL Files

Dereferencing the XML namespace defined in Section 1.6 XML Namespaces will produce the Resource Directory Description Language (RDDL) [RDDL] document that describes this namespace, including the XML schema [XML Schema Part 1, 2] and WSDL [WSDL 1.1] declarations associated with this specification.

SOAP bindings for the WSDL [WSDL 1.1], referenced in the RDDL [RDDL] document, MUST use “document” for the *style* attribute.

1.8 Normative References

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

[IANA] *Port Numbers*, http://www.iana.org/assignments/port-numbers, February 2005.

[Namespaces in XML 1.1] W3C Recommendation, *Namespaces in XML 1.1,* http://www.w3.org/TR/2004/REC-xml-names11-20040204/, 4 February 2004.

[RFC 2141] R. Moats, *URN Syntax*, http://www.ietf.org/rfc/rfc2141.txt, IETF RFC 2141, May 1997.

[RFC 2253] M. Wahl, et al, *Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names*, http://www.ietf.org/rfc/rfc2253.txt, IETF RFC 2253, December 1997.

[RFC 2255] T. Howes, et al, *The LDAP URL Format*, http://www.ietf.org/rfc/rfc2255.txt, IETF RFC 2255, December 1997.

[RFC 3986] T. Berners-Lee, et al, *Uniform Resource Identifiers (URI): Generic Syntax*, http://www.ietf.org/rfc/rfc3986.txt, IETF RFC 3986, January 2005.

[SOAP 1.1] W3C Note, *Simple Object Access Protocol (SOAP) 1.1*, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, 08 May 2000.

[SOAP 1.2] W3C Recommendation, *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*, http://www.w3.org/TR/2007/REC-soap12-part1-20070427/, April 2007.

[SOAP 1.2, Part 2] W3C Recommendation, *SOAP Version 1.2 Part 2: Adjuncts Second Edition), Section 7: SOAP HTTP Binding*, http://www.w3.org/TR/2007/REC-soap12-part2-20070427, April 2007.

[SOAP/UDP] OASIS Committee Draft 01, *SOAP-over-UDP Version 1.1*, http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/cd-01/wsdd-soapoverudp-1.1-spec-cd-01.pdf, October 2008.

[RFC 4122] P. Leach, et al, *A Universally Unique IDentifier (UUID) URN Namespace*, http://www.ietf.org/rfc/rfc4122.txt, IETF RFC 4122, July 2005.

[RFC 5280] P. Leach, et al, *A Universally Unique IDentifier (UUID) URN Namespace*, http://www.ietf.org/rfc/rfc4122.txt, IETF RFC 4122, July 2005.

[WS-Addressing] W3C Member Submission, *Web Services Addressing (WS-Addressing)*, http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/,10 August 2004.

[WS-SecureConversation] S. Anderson, et al, *Web Services Secure Conversation Language (WS-SecureConversation)*, http://schemas.xmlsoap.org/ws/2005/02/, February 2005.

[WS-Trust] S. Anderson, et al, *Web Services Trust Language (WS-Trust)*, http://schemas.xmlsoap.org/ws/2005/02/trust, February 2005.

[WS-Security] A. Nadalin, et al, *Web Services Security: SOAP Message Security*, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf, March 2004.

[RDDL] Jonathan Borden, et al, *Resource Directory Description Language (RDDL) 2.0*, http://www.openhealth.org/RDDL/20040118/rddl-20040118.html, 18 January 2004.

[WSDL 1.1] W3C Note, *Web Services Description Language (WSDL) 1.1*, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 15 March 2001.

[XML Schema, Part 1] W3C Recommendation, *XML Schema Part 1: Structures*, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/, 2 May 2001.

[XML Schema, Part 2] W3C Recommendation, *XML Schema Part 2: Datatypes*, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/, 02 May 2001.

[XML Sig] W3C Recommendation, *XML-Signature Syntax and Processing*, http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/, 12 February 2002.

2 Model

2.1 Endpoint References

As part of the discovery process, Target Services present to the network (a) a stable identifier and (b) one or more transport addresses at which network messages can be directed. This information is contained in an `a:EndpointReference` element [WS-Addressing]. Nearly all of the SOAP messages defined herein contain the `a:EndpointReference` element, a facsimile is reproduced here for convenience:

```
<a:EndpointReference>
  <a:Address>xs:anyURI</a:Address>
 [<a:ReferenceProperties> ... </a:ReferenceProperties>]?
  ...
</a:EndpointReference>
```

The combination of `a:Address` and `a:ReferenceProperties` provide a stable and globally-unique identifier.

Of particular interest is the required `a:Address` child element, which WS-Addressing specifies to contain either "a logical address or identifier", and does not require it to be a network-resolvable transport address. By convention, this specification recommends using a globally-unique identifier (GUID) based URN [RFC 4122] scheme URI in this element; if the value of this element is not a network-resolvable transport address, such transport address(es) are conveyed in a separate `d:XAddrs` element defined herein (see below).

2.2 Operational Modes

2.2.1 Ad hoc Mode

In an ad hoc mode discovery messages are sent multicast and response messages are sent unicast. Figure 1 depicts the message exchanges between a Target Service and a Client operating in an ad hoc mode.

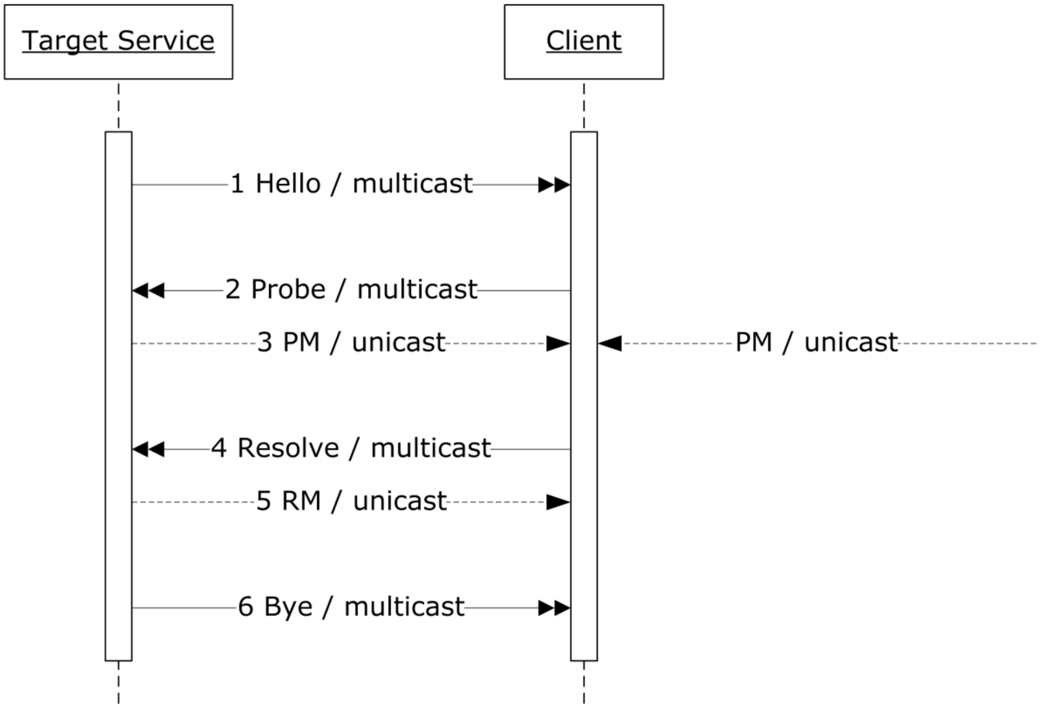


Figure 1 : Message Exchanges in an ad hoc mode.

A Target Service sends a multicast Hello message (1) when it joins a network (see Section 4.1.1 Target Service). A Client listens for multicast Hello messages (see Section 4.1.2 Client). A Client sends a multicast Probe message (2) to locate Target Services (see Section 5.2.1 Client). If a Target Service matches the Probe it responds with a unicast Probe Match (PM) message (3) (see Section 5.3.1 Target Service). Other matching Target Services may also send unicast Probe Match. A Client sends a multicast Resolve message (4) to locate a particular Target Service (see Section 6.2.1 Client). A Target Service that match the Resolve responds with a unicast Resolve Match (RM) message (5) (see Section 6.3.1 Target Service). A Target Service makes an effort to send a multicast Bye message (6) when it leaves a network (see Section 4.2.1 Target Service). A Client listens for multicast Bye messages (see 4.2.2 Client).

Figure 2 depicts the message exchanges in an ad hoc mode when a Discovery Proxy is present on the network.

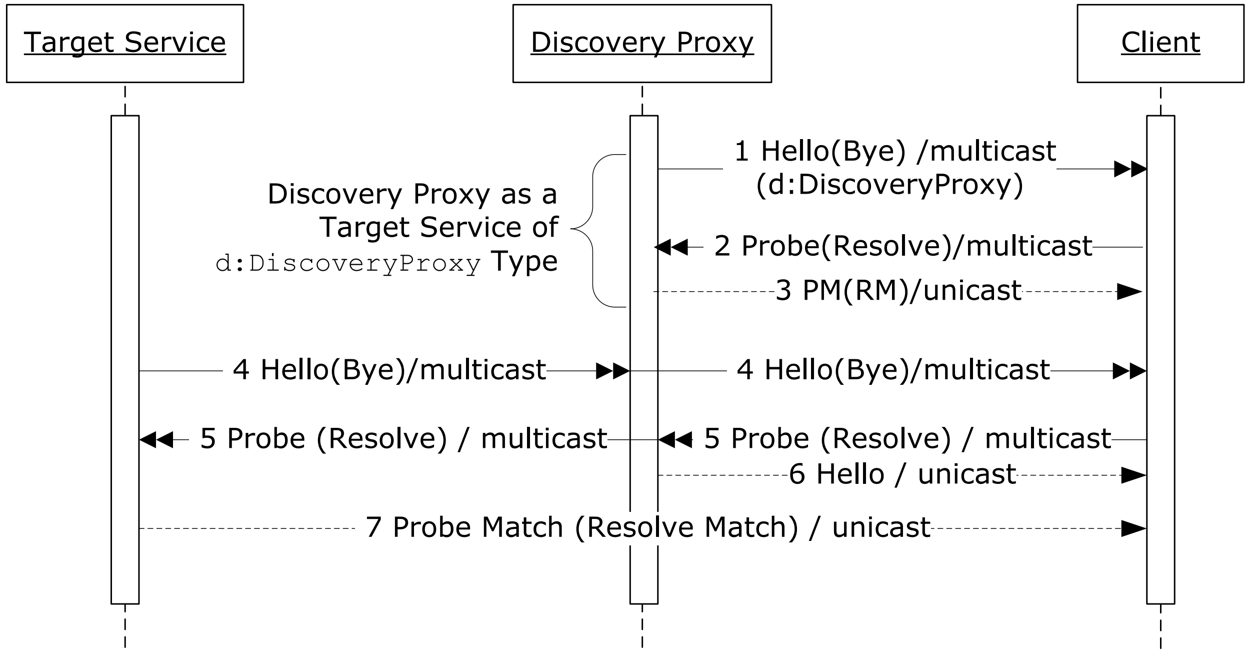
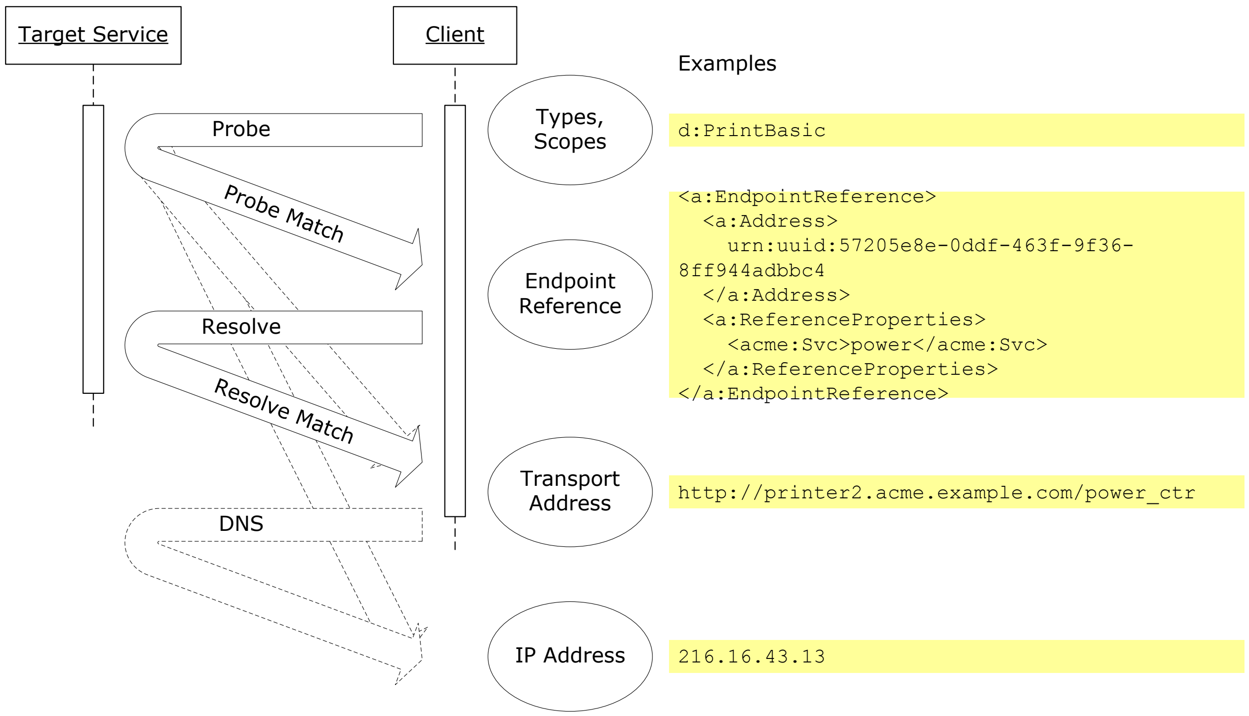

Target Service
Discovery Proxy
Client
1 Hello(Bye) /multicast (d:DiscoveryProxy)
Discovery Proxy as a Target Service of d:DiscoveryProxy Type
2 Probe(Resolve)/multicast
3 PM(RM)/unicast
4 Hello(Bye)/multicast
4 Hello(Bye)/multicast
5 Probe (Resolve) / multicast
5 Probe (Resolve) / multicast
6 Hello / unicast
7 Probe Match (Resolve Match) / unicast

Figure 2: Message exchanges in an ad hoc mode in the presence of a Discovery Proxy.

A Target Service sends multicast Hello and Bye (4) and responds to matching multicast Probe and Resolve (5,7). A Client listens for multicast Hello and Bye (4) and sends multicast Probe and Resolve (5). A Discovery Proxy is also a Target Service of a well known `d:DiscoveryProxy` type and sends a multicast Hello message annoucing its arrival on the network and; a multicast Bye message annoucing its depature from the network (1). It responds to the matching Probe and Resolve for itself (2), with a Probe Match (PM) and a Resolve Match (RM) respectively (3). If a Discovery Proxy is configured to reduce multicast traffic on the network, it listens for multicast Hello and Bye from other Target Services (4) and store/update information for corresponding Target Services (see Section 4.1.3 Discovery Proxy and 4.2.3 Discovery Proxy). It responds to the multicast Probe and Resolve for other Target Services (5), with a Hello message (6) (see Section 4.1.3 Discovery Proxy), indicating the Client to switch to managed mode and to sent unicast Probe and Resolve (see Section 2.2.2 Managed Mode).

Conceptually, Hello, Probe Match, and Resolve Match contain different kinds of information as Figure 3 depicts.



```
d:PrintBasic
```

```
<a:EndpointReference>
  <a:Address>
    urn:uuid:57205e8e-0ddf-463f-9f36-
8ff944adbbc4
  </a:Address>
  <a:ReferenceProperties>
    <acme:Svc>power</acme:Svc>
  </a:ReferenceProperties>
</a:EndpointReference>
```

```
http://printer2.acme.example.com/power_ctr
```

```
216.16.43.13
```

Figure 3 : Conceptual content of messages.

Starting at the top of Figure 3, Probe maps from Types and/or Scopes to an Endpoint Reference [WS-Addressing]; though not depicted, Hello also provides an Endpoint Reference. Resolve maps this information to one or more transport addresses. Other address mappings may be needed, e.g., DNS, but are beyond the scope of this specification.

The required components of each message are defined in detail below, but as an optimization, a Target Service may short-circuit these message exchanges by including additional components; for instance, a Probe Match may contain transport address(es) along with an Endpoint Reference, or a transport address may use an IP address instead of a DNS name.

2.2.2 Managed Mode

In a managed mode discovery messages are sent unicast to a Discovery Proxy. Figure 4 depicts the message exchanges between a Client, a Target Service and a Discovery Proxy in a managed mode.

A Target Service sends a unicast Hello message (1) to a Discovery Proxy when it joins a network (see Section 4.1.1 Target Service). A Client sends a unicast Probe request (2) to a Discovery Proxy to locate services (see Section 5.2.1 Client). A Discovery Proxy responds to a unicast Probe request with a Probe Match response (3) containing matching Target Services, if any (see Section 5.3.2 Discovery Proxy). A Client sends a unicast Resolve request (4) to a Discovery Proxy to locate a particular Target Service (see Section 6.2.1 Client). A Discovery Proxy respond to a unicast Resolve request with a Resolve Match response (4) containing the matching Target Service, if any (see Section 6.3.2 Discovery Proxy). A Target Service makes an effort to send a unicast Bye message (6) to a Discovery Proxy when it leaves a network (see Section 4.2.1 Target Service).

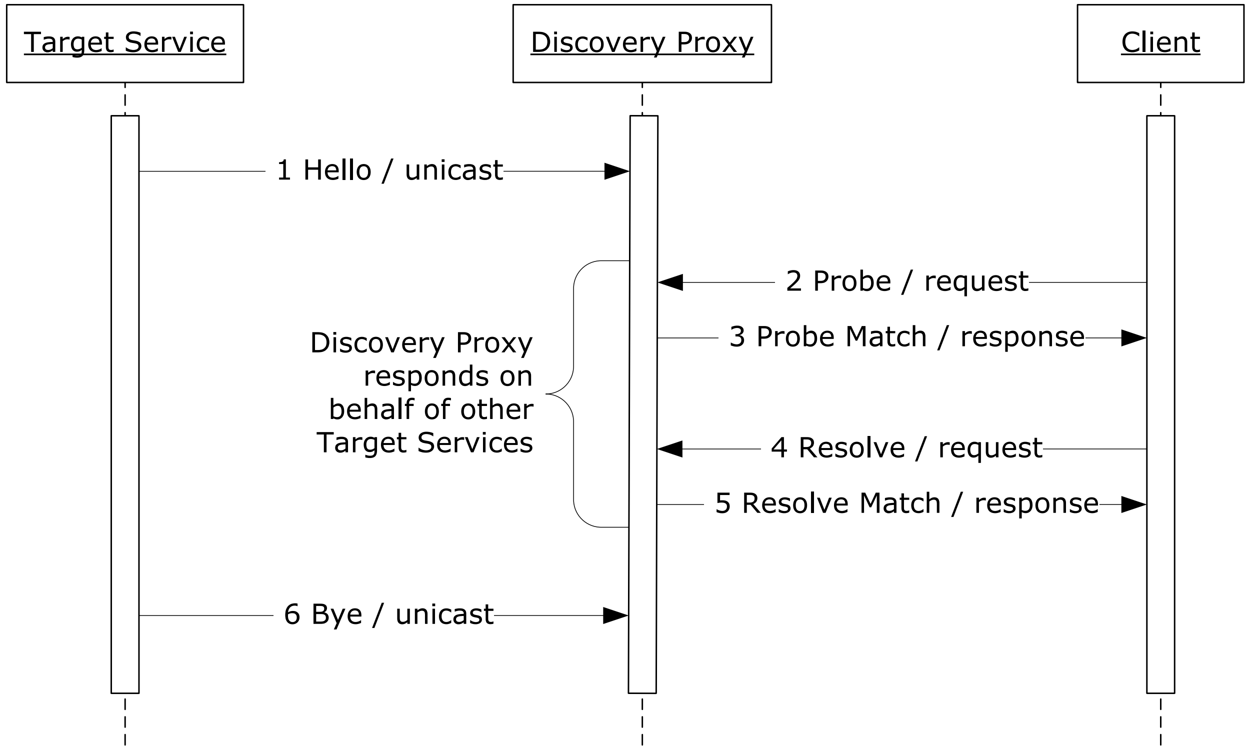


Figure 4: Message exchanges in a managed mode.

To operate in a managed mode a Target Service and a Client needs an Endpoint Reference of the Discovery Proxy. A Target Service or a Client can acquire this information from a number of ways including, but not not limited to; explicit configuration, explicit Probe for Discovery Proxy, DNS or DHCP, specifics of which are outside the scope of this specification. One such method that reduces the traffic in an ad hoc network and allows Client to dynamically switch to managed mode is described below.

2.2.3 Dynamic Mode Switching

To limit multicast traffic, Clients may be configured to dynamically switch from an ad hoc mode to a managed mode and vice versa, depicted in Figure 5.

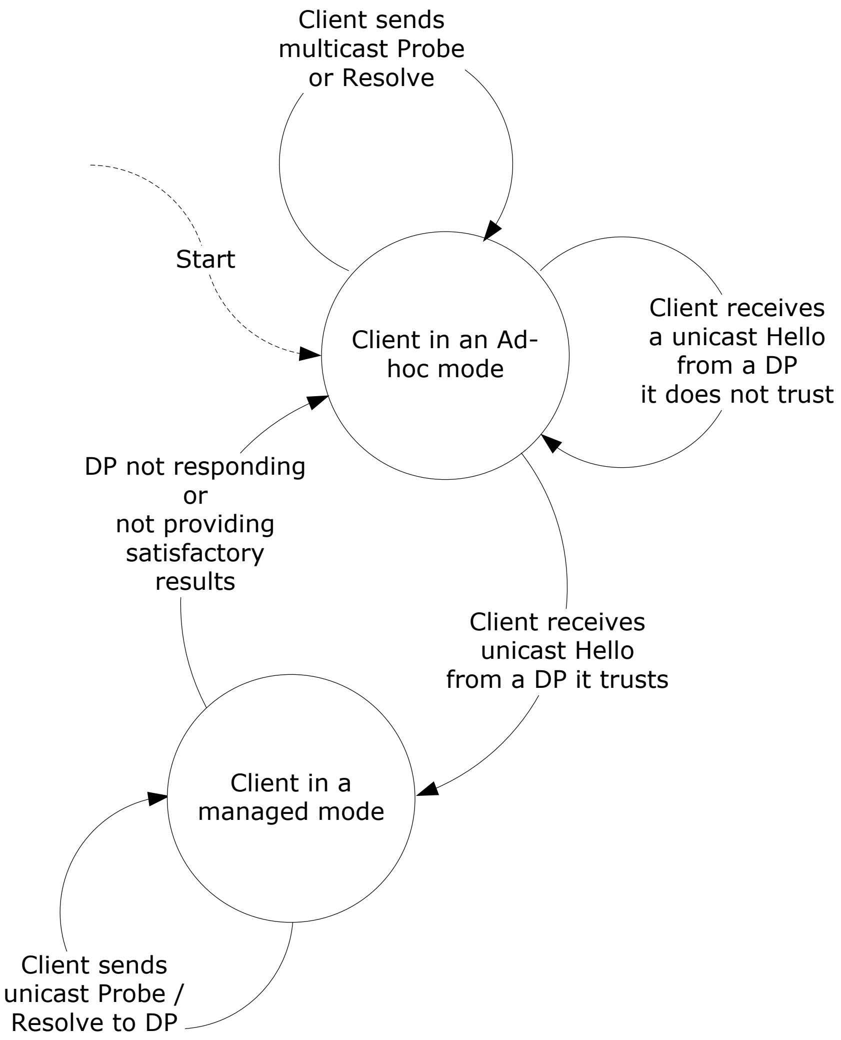


Figure 5: State transitions of a Client configured to dynamically switch operational modes.

By default, a Client assumes that no Discovery Proxy (DP) is available as a Discovery Proxy is an optional component and may not be present on the network. The Client operates in an ad hoc mode and listens for multicast Hello and Bye announcements, sends multicast Probe and/or Resolve messages, and listens for Probe Match and/or Resolve Match messages (see Section 2.2.1 Ad hoc Mode).

However, if one or more DP are available, those DP send a unicast Hello with a well-known "discovery proxy" type `d:DiscoveryProxy` in response to any multicast Probe or Resolve. As depicted in Figure 5, Clients listen for this signal that one or more DP are available, and for subsequent searches switches to a managed mode and instead of multicast, send Probe and Resolve messages unicast to one or more DP they trust whilst ignoring multicast Hello and Bye from Target Services.

In a managed mode, a Client communicates with a DP as described in Section 2.2.2 Managed Mode; using the transport information contained in the DP Hello; this is typically indicated by the scheme of a transport URI, e.g., "http:" (HTTP), "soap.udp:" (UDP [SOAP/UDP]), or other.

If the DP is unresponsive after DP_MAX_TIMEOUT, or if the Client finds the responses from the DP unsatisfactory, Clients revert to using the multicast messages specified herein.

Table 4 specifies the default value for this parameter.

Table 4: Default value for Discovery Proxy timeout parameter.

Parameter	Default Value
DP_MAX_TIMEOUT	5 seconds

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 multicast 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. Some Clients (for example, mobile clients frequently moving within and beyond managed environments) may be configured to Probe first for a DP and, only if such Probe fails, switch to the operational mode described above. Specific means of such configuration is beyond of the scope of this specification.

Unlike a Client, a Target Service operating in an ad hoc mode always sends (multicast) Hello and Bye, and always responds to Probe and Resolve with (unicast) Probe Match and Resolve Match respectively. A Target Service does not need to explicitly recognize and/or track the availability of a DP in an ad hoc mode – a Target Service behaves the same way in an ad hoc mode regardless of the presence or absence of a DP. This is because the Hello and Bye are too infrequent and therefore generate too little multicast traffic to warrant adding complexity to Target Service behavior. However, some Target Services may be configured to operate only in a managed mode and unicast Hello and Bye directly to a DP; these would not multicast Hello and Bye or respond to Probe or Resolve; specific means of such configuration are beyond the scope of this specification.

3 Protocol Assignments

3.1.1 Ad hoc mode over IP multicast

If IP multicast is used to send multicast messages described herein, they MUST be sent using the following assignments:

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

Other address bindings may be defined but are beyond the scope of this specification.

Messages sent over UDP MUST be sent using SOAP over UDP [SOAP/UDP]. To compensate for possible UDP unreliability, senders MUST use the example transmission algorithm in Appendix I of SOAP over UDP. In order to improve interoperability and network efficiency use of SOAP 1.2 protocol [SOAP 1.2] is RECOMMENDED.

3.1.2 Managed mode over HTTP

If the messages described herein are sent unicast using HTTP protocol, they MUST be sent using SOAP HTTP Binding as defined in Section 7 of SOAP 1.2 Part 2 [SOAP 1.2 Part 2].

3.1.3 Application Level Transmission Delay

As designated below, before sending some message types defined herein, a Target Service MUST wait for a timer to elapse before sending the message using the bindings described above. 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 value in Table 5 MAY be revised by other specifications.

Note: The authors expect this parameter to be adjusted based on interoperability test results.

Other transport bindings may be defined but are beyond the scope of this specification.

4 Hello and Bye

Support for messages described in this section MUST be implemented by a Target Service, MUST be implemented by a Discovery Proxy, and MAY be implemented by a Client as described below.

4.1 Hello

Hello message is sent by a Target Service to announce its availability when it joins the network. It is also sent by a Discovery Proxy to reduce multicast traffic on an ad hoc network.

The normative outline for Hello is:

```
<s:Envelope ... >
  <s:Header ... >
    <a:Action ... >
      http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/Hello
    </a:Action>
    <a:MessageID ... >xs:anyURI</a:MessageID>
   [<a:RelatesTo>
      xs:anyURI
    </a:RelatesTo>]?
    <a:To ... >urn:docs-oasis-open-org:ws-dd:discovery:2008:09</a:To>
    [<d:AppSequence ... />]?
    ...
  </s:Header>
  <s:Body ... >
    <d:Hello ... >
      <a:EndpointReference> ... </a:EndpointReference>
     [<d:Types>list of xs:QName</d:Types>]?
     [<d:Scopes>list of xs:anyURI</d:Scopes>]?
     [<d:XAddrs>list of xs:anyURI</d:XAddrs>]?
      <d:MetadataVersion>xs:unsignedInt</d:MetadataVersion>
      ...
    </d:Hello>
  </s:Body>
</s:Envelope>
```

The following describes additional normative constraints on the outline listed above:

/s:Envelope/s:Header/*

Per SOAP [SOAP 1.1, SOAP 1.2], header blocks MAY appear in any order.

/s:Envelope/s:Header/a:RelatesTo

MUST be included only by a Discovery Proxy and if and only if Hello is sent unicast in response to a multicast Probe (or Resolve). It MUST be the value of the **[message id]** property [WS-Addressing] of the multicast Probe (Resolve).

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

MUST be included.

In an ad hoc mode, it MUST be "`urn:docs-oasis-open-org:ws-dd:discovery:2008:09`" [RFC 2141] .

In a managed mode, it MUST be the [**address**] property [WS-Addressing] of the Endpoint Reference of the Discovery Proxy.

/s:Envelope/s:Header/d:AppSequence

MUST be included to allow ordering discovery messages from a Target Service (see Section 7 Application Sequencing).

SHOULD be omitted in a managed mode.

/s:Envelope/s:Body/*/a:EndpointReference

Endpoint Reference for the Target Service (or Discovery Proxy) (see Section 2.1 Endpoint References).

/s:Envelope/s:Body/*/d:Types

Unordered set of Types implemented by the Target Service (or Discovery Proxy).

- For a Target Service, if omitted or empty, no implied value. In a managed mode, all supported Types SHOULD be included.
- For a Discovery Proxy, MUST be included and MUST explicitly include `d:DiscoveryProxy`.

/s:Envelope/s:Body/*/d:Scopes

Unordered set of Scopes the Target Service (or Discovery Proxy) is in, which MAY be of more than one URI scheme. If included, MUST be a set of absolute URIs, and contained URIs MUST NOT contain white space. If omitted or empty, implied value is a set that includes "`http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/adhoc`".

In a managed mode, all Scopes SHOULD be included.

/s:Envelope/s:Body/*/d:XAddrs

Transport address(es) that MAY be used to communicate with the Target Service (or Discovery Proxy). Contained URIs MUST NOT contain white space.

In a managed mode, all transport address(es) SHOULD be included.

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

Incremented by >= 1 whenever there is a change in the metadata of the Target Service. If a Target Service goes down and comes back up again, this value MAY be incremented but MUST NOT be decremented (see Section 7 Application Sequencing). Metadata includes, but is not limited to, `../d:Types` and `../d:Scopes`. By design, this value MAY be used by the Client and/or Discovery Proxy for cache control of Target Service metadata.

4.1.1 Target Service

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` above).

To minimize the risk of a network storm and to not overwhelm the recipient (e.g., after a network crash and recovery or power blackout and restoration), a Target Service MUST wait for a timer to elapse before sending the Hello as described in Section 3.1.3 Application Level Transmission Delay.

In an ad hoc mode,

- A Hello MUST be sent multicast to "`urn:docs-oasis-open-org:ws-dd:discovery:2008:09`" [RFC 2141] .
- 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 (or a Discovery Proxy) may receive two such messages containing the same `/s:Envelope/s:Body/*/d:MetadataVersion` but containing different metadata. If a Client (or a Discovery Proxy) 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 (or a Discovery Proxy) SHOULD NOT delete cached metadata and SHOULD NOT repeat a Probe (or Resolve) if it detects differences in contained metadata.

Table 6 lists an example Hello sent multicast in an ad hoc mode by the same Target Service that responded with a Probe Match in Table 2.

Table 6: Example Hello sent multicast in an ad hoc mode

```
(01) <s:Envelope
(02)   xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(03)   xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09"
(04)   xmlns:s="http://www.w3.org/2003/05/soap-envelope" >
(05)   <s:Header>
(06)     <a:Action>
(07)       http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/Hello
(08)     </a:Action>
(09)     <a:MessageID>
(10)       urn:uuid:73948edc-3204-4455-bae2-7c7d0ff6c37c
(11)     </a:MessageID>
(12)     <a:To>urn:docs-oasis-open-org:ws-dd:discovery:2008:09</a:To>
(13)     <d:AppSequence InstanceId="1077004800" MessageNumber="1" />
(14)   </s:Header>
(15)   <s:Body>
(16)     <d:Hello>
(17)       <a:EndpointReference>
(18)         <a:Address>
(19)           urn:uuid:98190dc2-0890-4ef8-ac9a-5940995e6119
(20)         </a:Address>
(21)       </a:EndpointReference>
(22)       <d:MetadataVersion>75965</d:MetadataVersion>
(23)     </d:Hello>
(24)   </s:Body>
(25) </s:Envelope>
(26)
```

Lines (06-08) indicate this is a Hello, and because Line (12) is set to the distinguished URI defined herein, this is a multicast Hello. Line (13) contains an instance identifier as well as a message number; this information allows the receiver to reorder Hello and Bye messages from a Target Service. Lines (17-21) are identical to the corresponding lines in the Probe Match in Table 2.

In a managed mode,

- A Hello MUST be sent unicast to [**address**] property [WS-Addressing] of the Endpoint Reference of the Discovery Proxy.
- A Target Service SHOULD include complete metadata information in the Hello message.

Table 7 lists an example Hello sent unicast in a managed mode to a Discovery Proxy.

Table 7: Example Hello sent unicast in a managed mode to a Discovery Proxy

```
(01) <s:Envelope
(02)   xmlns:a="http://www.w3.org/2005/08/addressing"
(03)   xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09"
(04)   xmlns:s="http://www.w3.org/2003/05/soap-envelope">
(05)   <s:Header>
(06)     <a:Action>
(07)       http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/Hello
(08)     </a:Action>
(09)     <a:MessageID>
(10)       urn:uuid:b10688d7-ea05-4bb1-a6bc-3aaf3be47f8e
(11)     </a:MessageID>
(12)     <a:To>http://example.com/DiscoveryProxy</a:To>
```

```
(13)   </s:Header>
(14)   <s:Body>
(15)     <d:Hello>
(16)       <a:EndpointReference>
(17)         <a:Address>
(18)           urn:uuid:98190dc2-0890-4ef8-ac9a-5940995e6119
(19)         </a:Address>
(20)       </a:EndpointReference>
(21)       <d:Types>i:PrintBasic i:PrintAdvanced</d:Types>
(22)       <d:Scopes>
(23)         ldap:///ou=engineering,o=exampleorg,c=us
(24)         ldap:///ou=floor1,ou=b42,ou=anytown,o=exampleorg,c=us
(25)         http://itdept/imaging/deployment/2004-12-04
(26)       </d:Scopes>
(27)       <d:XAddrs>http://prn-example/PRN42/b42-1668-a</d:XAddrs>
(28)       <d:MetadataVersion>75965</d:MetadataVersion>
(29)     </d:Hello>
(30)   </s:Body>
(31) </s:Envelope>
(32)
```

Lines (06-08) indicate this is a Hello, and Line (12) indicates it is sent unicast to Discovery Proxy over HTTP. The AppSequence header is omitted here because the messages sent over HTTP are received in the same order in which they are sent. The Lines (16-28) describe a single Target Service and they are identical to corresponding lines (24-36) in the Probe Match in Table 2. This Hello message sent in a managed mode contains complete information, Lines (16-28); about the Target Service, as opposed to the one sent in the ad hoc mode, Lines (17-22) in Table 6.

4.1.2 Client

In an ad hoc mode,

- To minimize the need to Probe, Clients SHOULD listen for Hello messages and store (or update) information for the corresponding Target Services.
- If a Client receives a Hello message from a Discovery Proxy in response to a multicast Probe (or Resolve) (see Section 4.1.3 Discovery Proxy), the Client SHOULD switch to a managed mode and send unicast Probe (or Resolve) to the Discovery Proxy (see Section 2.2.3 Dynamic Mode Switching).

4.1.3 Discovery Proxy

In an ad hoc mode,

- A Discovery Proxy MUST send a Hello for itself (as a Target Service of `d:DiscoveryProxy` type) as described in Section 4.1.1 Target Service.
- A Discovery Proxy MAY be configured to reduce multicast traffic on an ad hoc network, in this capacity:
  - A Discovery Proxy MUST listen for multicast Hello messages and store (or update) information for the corresponding Target Services.
  - A Discovery Proxy MUST listen for multicast Probe (and Resolve). In response to any multicast Probe (or multicast Resolve) from a Client, a Discovery Proxy MUST send a unicast Hello to the Client and SHOULD send the Hello without waiting for a timer to elapse.

In a managed mode,

- A Discovery Proxy MUST listen for unicast Hello messages and store (or update) information for the corresponding Target Services.

4.2 Bye

Bye message is sent by a Target Service when it is preparing to leave the network.

The normative outline for Bye is:

```
<s:Envelope ... >
  <s:Header ... >
    <a:Action ... >
      http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/Bye
    </a:Action>
    <a:MessageID ... >xs:anyURI</a:MessageID>
    <a:To ...>urn:docs-oasis-open-org:ws-dd:discovery:2008:09</a:To>
    [<d:AppSequence ... />]?
    ...
  </s:Header>
  <s:Body ... >
    <d:Bye ... >
      <a:EndpointReference> ... </a:EndpointReference>
      ...
    </d:Bye>
  </s:Body>
</s:Envelope>
```

The following describes additional normative constraints on the outline listed above:

/s:Envelope/s:Header/*

Per SOAP [SOAP 1.1, SOAP 1.2], header blocks MAY appear in any order.

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

As constrained for Hello (see Section 4.1 Hello).

/s:Envelope/s:Header/d:AppSequence

As constrained for Hello (see Section 4.1 Hello).

/s:Envelope/s:Body/*/a:EndpointReference

Endpoint Reference for the Target Service (see Section 2.1 Endpoint References).

4.2.1 Target Service

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 to elapse.

In an ad hoc mode,

- A Bye MUST be sent multicast to "`urn:docs-oasis-open-org:ws-dd:discovery:2008:09`" [RFC 2141].

Table 8 lists an example Bye message sent multicast in an ad hoc mode corresponding to the Hello in Table 6.

Table 8 Example Bye message sent multicast in an ad hoc mode.

```
<s:Envelope
    xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09"
    xmlns:s="http://www.w3.org/2003/05/soap-envelope" >
  <s:Header>
    <a:Action>
      http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/Bye
    </a:Action>
    <a:MessageID>
      urn:uuid:337497fa-3b10-43a5-95c2-186461d72c9e
    </a:MessageID>
    <a:To>urn:docs-oasis-open-org:ws-dd:discovery:2008:09</a:To>
    <d:AppSequence InstanceId="1077004800" MessageNumber="4" />
  </s:Header>
  <s:Body>
```

```
(16)     <d:Bye>
(17)       <a:EndpointReference>
(18)         <a:Address>
(19)           urn:uuid:98190dc2-0890-4ef8-ac9a-5940995e6119
(20)         </a:Address>
(21)       </a:EndpointReference>
(22)     </d:Bye>
(23)   </s:Body>
(24) </s:Envelope>
(25)
```

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.

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.

In a managed mode,

- A Bye MUST be sent unicast to [**address**] property [WS-Addressing] of the Endpoint Reference of the Discovery Proxy.

Table 9 lists an example Bye message corresponding to the Hello message in Table 7, sent unicast in a managed mode to a Discovery Proxy.

Table 9: Example Bye message sent unicast in a managed mode to a Discovery Proxy.

```
(01) <s:Envelope
(02)   xmlns:a="http://www.w3.org/2005/08/addressing"
(03)   xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09"
(04)   xmlns:s="http://www.w3.org/2003/05/soap-envelope">
(05)   <s:Header>
(06)     <a:Action>
(07)       http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/Bye
(08)     </a:Action>
(09)     <a:MessageID>
(10)       urn:uuid:cceb5804-1bcc-4721-bef3-dd688763b6aa
(11)     </a:MessageID>
(12)     <a:To>http://example.com/DiscoveryProxy</a:To>
(13)   </s:Header>
(14)   <s:Body>
(15)     <d:Bye>
(16)       <a:EndpointReference>
(17)         <a:Address>
(18)           urn:uuid:98190dc2-0890-4ef8-ac9a-5940995e6119
(19)         </a:Address>
(20)       </a:EndpointReference>
(21)     </d:Bye>
(22)   </s:Body>
(23) </s:Envelope>
(24)
```

Lines (06-08) indicate this is a Bye, and like Hello in Table 7, Line (12) indicate that it is sent unicast to a Discovery Proxy over HTTP. Like Hello in Table 7, the application sequencing information is omitted because the messages sent unicast over HTTP are received in the same order in which they are sent. Like Bye in Table 10 the Body (Lines 15-21) is an abbreviated form of the corresponding information in the Hello.

4.2.2 Client

In an ad hoc mode, 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.

4.2.3 Discovery Proxy

In an ad hoc mode,

- A Discovery Proxy SHOULD send a Bye for itself (as a Target Service of `d:DiscoveryProxy` type) when it is preparing to leave the network as described in Section 4.2.1 Target Service.
- A Discovery Proxy MAY be configured to reduce multicast traffic on an ad hoc network, in this capacity:
  - A Discovery Proxy MUST listen for multicast Bye messages, marking or removing corresponding information as invalid.

In a managed mode,

- A Discovery Proxy MUST listen for unicast Bye messages, marking or removing corresponding information as invalid.

Note that both in an ad hoc mode and a managed mode, a Discovery Proxy MAY retain information associated with a Target Service that has left the network, for instance if the Discovery Proxy expects the Target Service to rejoin the network at some point in the future. Conversely, Discovery Proxy 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.

5 Probe and Probe Match

To find Target Services by the Type of the Target Service, a Scope in which the Target Service resides, both, or simply all Target Services, a Client sends a Probe.

Support for messages described in this section MUST be implemented by a Target Service, MUST be implemented by a Discovery Proxy, and MAY be implemented by a Client as described below.

5.1 Matching Types and Scopes

A Probe includes zero, one, or two constraints on matching Target Services: a set of Types and/or a set of Scopes. A Probe Match MUST include a Target Service if and only if all of the Types and all of the Scopes in the Probe match the Target Service.

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

- The namespace [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 of a Target Service per the rule indicated within the Probe. This specification defines the following matching rules. 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 regardless of the value of S1 and/or S2.

`http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/rfc3986`

Using a case-insensitive comparison,

- The `scheme` [RFC 3986] of S1 and S2 is the same and
- The `authority` of S1 and S2 is the same and

Using a case-sensitive comparison,

- The `path_segments` of S1 is a `segment`-wise (not string) prefix of the `path_segments` of S2 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 matching rule does NOT test whether the string representation of S1 is a prefix of the string representation of S2. For example, "http://example.com/abc" matches "http://example.com/abc/def" using this rule but "http://example.com/a" does not.

`http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/uuid`

Using a case-insensitive comparison, the scheme of S1 and S2 is "uuid" and each of the unsigned integer fields [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://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/ldap`

Using a case-insensitive comparison, the scheme of S1 and S2 is "ldap" and the hostport [RFC 2255] of S1 and S2 is the same and the RDNSequence [RFC 2253] of the dn of S1 is a prefix of the RDNSequence of the dn of S2, where comparison does not support the variants in an RDNSequence described in Section 4 of RFC 2253 [RFC 2253].

`http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/strcmp0`

Using a case-sensitive comparison, the string representation of S1 and S2 is the same.

5.2 Probe

The normative outline for Probe is:

```
<s:Envelope ... >
  <s:Header ... >
    <a:Action ... >
      http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/Probe
    </a:Action>
    <a:MessageID ... >xs:anyURI</a:MessageID>
   [<a:ReplyTo ... >endpoint-reference</a:ReplyTo>]?
    <a:To ... >xs:anyURI</a:To>
    ...
  </s:Header>
  <s:Body ... >
    <d:Probe ... >
     [<d:Types>list of xs:QName</d:Types>]?
     [<d:Scopes [MatchBy="xs:anyURI"]? ... >
        list of xs:anyURI
      </d:Scopes>]?
     ...
    </d:Probe>
  </s:Body>
</s:Envelope>
```

The following describes additional normative constraints on the outline listed above:

/s:Envelope/s:Header/*

Per SOAP [SOAP 1.1, SOAP 1.2], header blocks MAY appear in any order.

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

If included, MUST be of type a:EndpointReferenceType [WS-Addressing]. If omitted, implied value of the **[reply endpoint]** property [WS-Addressing] is "`http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous`".

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

If the value is "`http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous`", **[reply endpoint]** property is defined by the underlying transport. For example, if the Probe was received over UDP using the assignments listed in Section 3.1.1 Ad hoc mode over IP multicast, the **[reply endpoint]** is the IP source address and port number of the Probe transport header [SOAP/UDP].

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

- If sent to a Target Service, MUST be "`urn:docs-oasis-open-org:ws-dd:discovery:2008:09`" [RFC 2141].
- If sent to a Discovery Proxy, MUST be the **[address]** property of the Endpoint Reference for the Discovery Proxy, e.g., as contained in a Hello from the Discovery Proxy.

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

If omitted or empty, implied value is any Type.

/s:Envelope/s:Body/d:Probe/d:Scopes

If included, MUST be a list of absolute URIs, and contained URIs MUST NOT contain whitespace. The contained URIs MAY be of more than one URI scheme. If omitted or empty., implied value is any Scope.

/s:Envelope/s:Body/d:Probe/d:Scopes/@MatchBy

If omitted, implied value is

" `http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/rfc3986` ".

The value MUST be compared per RFC 3986 Section 6.2.1 Simple String Comparison [RFC 3986].

If a Target Service or Discovery Proxy receives a unicast Probe and does not support the matching rule, it MAY choose not to send a Probe Match and instead generate a fault, bound to SOAP [WS-Addressing] as follows:

[action]	http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/fault
[Code]	s12:Sender
[Subcode]	d:MatchingRuleNotSupported
[Reason]	E.g., the matching rule specified is not supported.

[Detail]	`<d:SupportedMatchingRules>`
	  `list of xs:anyURI`
	`</d:SupportedMatchingRules>`

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

5.2.1 Client

A Client MAY send a Probe to find Target Services of a given Type and/or in a given Scope or to find Target Services regardless of their Types or Scopes.

In an ad hoc mode,

- A Probe is a one-way message.
- A Probe MUST be sent multicast to "`urn:docs-oasis-open-org:ws-dd:discovery:2008:09`" [RFC 2141] .

In an ad hoc network a Client may not know in advance how many Target Services (if any) will send Probe Match, the Client MAY adopt either of the following behaviors:

- Wait for a sufficient number of Probe Match messages.
- Repeat the Probe several times until the Client is convinced that no further Probe Match messages will be received. The Client MUST use the same value for the **[message id]** property [WS-Addressing] in all copies of the Probe.

If a Client knows a transport address of a Target Service, the Probe MAY be sent unicast to that address.

Table 1 lists an example Probe message sent multicast by a Client searching for a printer in an ad hoc mode.

In a managed mode,

- A Probe is a request message.
- A Probe MUST be sent unicast to [**address**] property [WS-Addressing] of the Endpoint Reference of the Discovery Proxy.

Table 10 lists an example Probe message sent unicast to a Discovery Proxy by a Client searching for a printer in a managed mode.

Table 10: Example Probe sent unicast to a Discovery Proxy in a managed mode.

```
<s:Envelope
    xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09"
    xmlns:i="http://printer.example.org/2003/imaging"
    xmlns:s="http://www.w3.org/2003/05/soap-envelope" >
  <s:Header>
    <a:Action>
      http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/Probe
    </a:Action>
    <a:MessageID>
      urn:uuid:d78c2d8d-1123-4a51-a814-955efdded812
    </a:MessageID>
```

```
(13)     <a:To>http://example.com/DiscoveryProxy</a:To>
(14)   </s:Header>
(15)   <s:Body>
(16)     <d:Probe>
(17)       <d:Types>i:PrintBasic</d:Types>
(18)       <d:Scopes
(19)    MatchBy="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/ldap" >
(20)         ldap:///ou=engineering,o=examplecom,c=us
(21)       </d:Scopes>
(22)     </d:Probe>
(23)   </s:Body>
(24) </s:Envelope>
(25)
```

Lines (07-09) in Table 10 indicate this message is a Probe, and Line (13) indicates it is being sent to a Discovery Proxy over HTTP.

Lines (17-21) specify two constants on the Target Services and they are identical to the corresponding Lines (17-21) in Table 1.

5.2.2 Target Service

In an ad hoc mode,

- A Target Service MUST listen for multicast Probe messages and respond as described in Section 5.3.1 Target Service.
- A Target Service MAY listen for unicast Probe requests and respond to them as described in Section 5.3.1 Target Service.

5.2.3 Discovery Proxy

In an ad hoc mode,

- A Discovery Proxy MUST listen for multicast Probe messages for itself and respond as described in Section 5.3.2 Discovery Proxy.
- A Discovery Proxy MAY be configured to reduce multicast traffic on an ad hoc network, in this capacity, a Discovery Proxy MUST listen for multicast Probe for other Target Services and respond to them with a Hello message as described in Section 4.1.3 Discovery Proxy.

In a managed mode,

- A Discovery Proxy MUST listen for unicast Probe request and respond to them as described in Section 5.3.2 Discovery Proxy.

5.3 Probe Match

Probe Match is sent by a Target Service or a Discovery Proxy in response to a Probe.

The normative outline for Probe Match is:

```
<s:Envelope ... >
  <s:Header ... >
    <a:Action ... >
      http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/ProbeMatches
    </a:Action>
    <a:MessageID ... >xs:anyURI</a:MessageID>
    <a:RelatesTo ... >xs:anyURI</a:RelatesTo>
    <a:To ... >xs:anyURI</a:To>
    [<d:AppSequence ... />]?
    ...
  </s:Header>
  <s:Body ... >
    <d:ProbeMatches ... >
     [<d:ProbeMatch ... >
        <a:EndpointReference> ... </a:EndpointReference>
```

```
       [<d:Types>list of xs:QName</d:Types>]?
       [<d:Scopes>list of xs:anyURI</d:Scopes>]?
       [<d:XAddrs>list of xs:anyURI</d:XAddrs>]?
        <d:MetadataVersion>xs:unsignedInt</d:MetadataVersion>
        ...
      </d:ProbeMatch>]*
      ...
    </d:ProbeMatches>
  </s:Body>
</s:Envelope>
```

The following describes additional normative constraints on the outline listed above:

/s:Envelope/s:Header/*

Per SOAP [SOAP 1.1, SOAP 1.2], header blocks MAY appear in any order.

/s:Envelope/s:Header/a:RelatesTo

MUST be the value of the **[message id]** property [WS-Addressing] of the Probe.

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

If the **[reply endpoint]** property [WS-Addressing] of the corresponding Probe is the IP source address and port number of the Probe transport header (e.g., when the a:ReplyTo header block was omitted from the corresponding Probe), the value of this header block MUST be "`http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous`".

/s:Envelope/s:Header/d:AppSequence

MUST be included to allow ordering discovery messages from a Target Service (see Section 7 Application Sequencing).

SHOULD be omitted in a managed mode.

/s:Envelope/s:Body/d:ProbeMatches

Matching Target Services.

- If this Probe Match was sent by a Target Service, this element will contain one `d:ProbeMatch` child. (If Target Service doesn't match the Probe, the Target Service does not send a Probe Match at all.)
- If this Probe Match was sent by a Discovery Proxy, this element will contain zero or more `d:ProbeMatch` children. (Discovery Proxies always respond to Probe.)

/s:Envelope/s:Body/d:ProbeMatches/d:ProbeMatch/a:EndpointReference

Endpoint Reference for the Target Service (see Section 2.1 Endpoint References).

/s:Envelope/s:Body/d:ProbeMatches/d:ProbeMatch/d:Types

See /s:Envelope/s:Body/*/d:Types in Section 4.1 Hello.

/s:Envelope/s:Body/d:ProbeMatches/d:ProbeMatch/d:Scopes

See /s:Envelope/s:Body/*/d:Scopes in Section 4.1 Hello.

/s:Envelope/s:Body/d:ProbeMatches/d:ProbeMatch/d:XAddrs

See /s:Envelope/s:Body/*/d:XAddrs in Section 4.1 Hello.

/s:Envelope/s:Body/d:ProbeMatches/d:ProbeMatch/d:MetadataVersion

See /s:Envelope/s:Body/*/d:MetadataVersion in Section 4.1 Hello.

5.3.1 Target Service

In an ad hoc mode,

- If a Target Service receives a Probe that match, it MUST respond with a Probe Match message. If the Target Service receives more than one copy of the Probe as determined by the **[message id]** property [WS-Addressing], it SHOULD respond only once. A Target Service MUST wait for a timer to elapse after receiving a Probe and before sending a Probe Match as described in Section 3.1.3

Application Level Transmission Delay. The Probe Match MUST be unicast to the **[reply endpoint]** property [WS-Addressing] of the Probe.

- If a Target Service receives a Probe and does not match the Probe, it MUST NOT respond with a Probe Match.

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

5.3.2 Discovery Proxy

In an ad hoc mode,

- If a Discovery Proxy receives a Probe for itself as determined by the presence of `d:DiscoveryProxy` in the Types, it MUST respond with a Probe Match message and MUST wait for a timer to elapse (see Section 3.1.3 Application Level Transmission Delay). The Probe Match MUST be unicast to the **[reply endpoint]** property [WS-Addressing] of the Probe.
- A Discovery Proxy MAY be configured to reduce multicast traffic on an ad hoc network, in this capacity, if a Discovery Proxy receives a Probe for other Target Services it MUST respond with a Hello (see Section 4.1.3 Discovery Proxy).

In a managed mode,

- If a Discovery Proxy receives a Probe request it MUST respond with a Probe Match message without waiting for a timer to elapse. The Probe Match SHOULD include complete metadata information about the matching Target Services. However, the Probe Match MAY contain zero matches if the Discovery Proxy has no matching Target Services.

Table 11 lists an example Probe Match message sent by the Discovery Proxy in response to the Probe message in Table 10.

Table 11: Example Probe Match sent in response to the managed Probe in Table 10

```
<s:Envelope
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09"
  xmlns:i="http://printer.example.org/2003/imaging"
  xmlns:s="http://www.w3.org/2003/05/soap-envelope" >
  <s:Header>
    <a:Action>
      http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/ProbeMatches
    </a:Action>
    <a:MessageID>
      urn:uuid:7e5bb4ee-621a-4ea6-b326-3db7d99ddb47
    </a:MessageID>
    <a:To>
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    </a:To>
    <a:RelatesTo>
      urn:uuid:d78c2d8d-1123-4a51-a814-955efdded812
    </a:RelatesTo>
  </s:Header>
  <s:Body>
    <d:ProbeMatches>
      <d:ProbeMatch>
        <a:EndpointReference>
          <a:Address>
            urn:uuid:98190dc2-0890-4ef8-ac9a-5940995e6119
          </a:Address>
        </a:EndpointReference>
        <d:Types>i:PrintBasic i:PrintAdvanced</d:Types>
        <d:Scopes>
          ldap:///ou=engineering,o=examplecom,c=us
          ldap:///ou=floor1,ou=b42,ou=anytown,o=examplecom,c=us
          http://itdept/imaging/deployment/2004-12-04
        </d:Scopes>
        <d:XAddrs>http://prn-example/PRN42/b42-1668-a</d:XAddrs>
```

```
(35)         <d:MetadataVersion>75965</d:MetadataVersion>
(36)       </d:ProbeMatch>
(37)       <d:ProbeMatch>
(38)         <a:EndpointReference>
(39)           <a:Address>
(40)             urn:uuid:70eda11c-200a-4a5e-b60e-d6793e77ace3
(41)           </a:Address>
(42)         </a:EndpointReference>
(43)         <d:Types>i:PrintBasic</d:Types>
(44)         <d:Scopes>
(45)           ldap:///ou=engineering,o=examplecom,c=us
(46)           ldap:///ou=floor1,ou=b42,ou=anytown,o=examplecom,c=us
(47)           http://itdept/imaging/deployment/2008-10-16
(48)         </d:Scopes>
(49)         <d:XAddrs>http://prn-example/PRN42/b42-1668-b</d:XAddrs>
(50)         <d:MetadataVersion>23654</d:MetadataVersion>
(51)       </d:ProbeMatch>
(52)     </d:ProbeMatches>
(53)   </s:Body>
(54) </s:Envelope>
(55)
```

Lines (07-09) in Table 11 indicate this message is a Probe Match; and Lines (13-15) indicate that it is a response to the Probe message in Table 10. Since this Probe Match message was sent over HTTP in response to the Probe message and since messages sent over HTTP are received in the order they are sent, it does not contain a header that identifies the instance number and message number like Line (19) in Table 2.

Lines (23-35) describe a Target Service and they are identical to the corresponding lines (24-36) in Table 2.

Lines (38-50) describe another Target Service, a basic printer service; that match the Probe in Table 10.

6 Resolve and Resolve Match

To locate a Target Service, i.e., to retrieve its transport address(es), a Client sends a Resolve.

Support for messages described in this section MUST be implemented by a Target Service, MUST be implemented by a Discovery Proxy and MAY be implemented by a Client as described below.

6.1 Matching Endpoint Reference

A Resolve includes a constraint on matching Target Service: an Endpoint Reference [WS-Addressing]. A Resolve Match MUST include a Target Service if and only if the Endpoint Reference in the Resolve match the Target Service per WS-Addressing Section 2.4 Endpoint Reference Comparison [WS-Addressing]

6.2 Resolve

The normative outline for Resolve is:

```
<s:Envelope ... >
  <s:Header ... >
    <a:Action ... >
      http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/Resolve
    </a:Action>
    <a:MessageID ... >xs:anyURI</a:MessageID>
   [<a:ReplyTo ... >endpoint-reference</a:ReplyTo>]?
    <a:To ... >xs:anyURI</a:To>
    ...
  </s:Header>
  <s:Body>
    <d:Resolve ... >
      <a:EndpointReference> ... </a:EndpointReference>
      ...
    </d:Resolve>
  </s:Body>
</s:Envelope>
```

The following describes additional normative constraints on the outline above:

/s:Envelope/s:Header/*

Per SOAP [SOAP 1.1, SOAP 1.2], header blocks MAY appear in any order.

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

As constrained for Probe (see Section 5.2 Probe).

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

As constrained for Probe (see Section 5.2 Probe).

/s:Envelope/s:Body/*/a:EndpointReference

Endpoint Reference for the Target Service (see Section 2.1 Endpoint References).

6.2.1 Client

A Client MAY send a Resolve to retrieve network transport information for a Target Service if it has an Endpoint Reference [WS-Addressing] for the Target Service.

In an ad hoc mode,

- A Resolve is a one-way message.
- A Resolve MUST be sent multicast to "urn:docs-oasis-open-org:ws-dd:discovery:2008:09" [RFC 2141] .

In a managed mode,

- A Resolve MUST be sent unicast to [**address**] property [WS-Addressing] of the Endpoint Reference of the Discovery Proxy.

6.2.2 Target Service

In an ad hoc mode,

- A Target Service MUST listen for multicast Resolve messages and respond to them as described in Section 6.3.1 Target Service.

6.2.3 Discovery Proxy

In an ad hoc mode,

- A Discovery Proxy MUST listen for multicast Resolve messages for itself and respond to them as described in Section 6.3.2 Discovery Proxy.
- A Discovery Proxy MAY be configured to reduce multicast traffic on an ad hoc network, in this capacity, a Discovery Proxy MUST listen for multicast Resolve for other Target Services and respond to them with a Hello message as described in Section 4.1.3 Discovery Proxy.

In a managed mode,

- A Discovery Proxy MUST listen for unicast Resolve requests and respond to them as described in Section 6.3.2 Discovery Proxy.

6.3 Resolve Match

Resolve Match is sent by a Target Service or a Discovery Proxy in response to a Resolve.

The normative outline for Resolve Match is:

```
<s:Envelope ... >
  <s:Header ... >
    <a:Action ... >
      http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/ResolveMatches
    </a:Action>
    <a:MessageID ... >xs:anyURI</a:MessageID>
    <a:RelatesTo ... >xs:anyURI</a:RelatesTo>
    <a:To ... >xs:anyURI</a:To>
    [<d:AppSequence ... />]?
    ...
  </s:Header>
  <s:Body ... >
    <d:ResolveMatches ... >
     [<d:ResolveMatch ... >
        <a:EndpointReference> ... </a:EndpointReference>
       [<d:Types>list of xs:QName</d:Types>]?
       [<d:Scopes>list of xs:anyURI</d:Scopes>]?
        <d:XAddrslist of xs:anyURI</d:XAddrs>
        <d:MetadataVersion>xs:unsignedInt</d:MetadataVersion>
        ...
      </d:ResolveMatch>]?
      ...
    </d:ResolveMatches>
  </s:Body>
</s:Envelope>
```

The following describes additional normative constraints on the outline listed above:

/s:Envelope/s:Header/*

Per SOAP [SOAP 1.1, SOAP 1.2], header blocks MAY appear in any order.

/s:Envelope/s:Header/a:RelatesTo

MUST be the value of the **[message id]** property [WS-Addressing] of the Resolve.

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

As constrained for Probe Match (see Section 5.3 Probe Match).

/s:Envelope/s:Header/d:AppSequence

As constrained for Probe Match (see Section 5.3 Probe Match).

/s:Envelope/s:Body/d:ResolveMatches

Matching Target Service.

/s:Envelope/s:Body/d:ResolveMatches/d:ResolveMatch/a:EndpointReference

Endpoint Reference for the Target Service (see Section 2.1 Endpoint References).

/s:Envelope/s:Body/d:ResolveMatches/d:ResolveMatch/d:Types

See /s:Envelope/s:Body/*/d:Types in Section 4.1 Hello.

/s:Envelope/s:Body/d:ResolveMatches/d:ResolveMatch/d:Scopes

See /s:Envelope/s:Body/*/d:Types in Section 4.1 Hello.

/s:Envelope/s:Body/d:ResolveMatches/d:ResolveMatch/d:XAddrs

See /s:Envelope/s:Body/*/d:Types in Section 4.1 Hello.

/s:Envelope/s:Body/d:ResolveMatches/d:ResolveMatch/d:MetadataVersion

See /s:Envelope/s:Body/*/d:Types in Section 4.1 Hello.

6.3.1 Target Service

In an ad hoc mode,

- If a Target Service receives a Resolve that matches it MUST respond with a Resolve Match message. If the Target Service receives more than one copy of the Resolve as determined by the **[message id]** property [WS-Addressing], it SHOULD respond only once. The Resolve Match MUST be unicast to the **[reply endpoint]** property [WS-Addressing] of the Resolve without waiting for a timer to elapse.
- If a Target Service receives a Resolve that does not match, it MUST NOT respond with a Resolve Match.

6.3.2 Discovery Proxy

In an ad hoc mode,

- If a Discovery Proxy receives a Resolve for itself, it MUST respond with a Resolve Match message. If the Discovery Proxy receives more than one copy of the Resolve as determined by the **[message id]** property [WS-Addressing], it SHOULD respond only once. The Resolve Match MUST be unicast to the **[reply endpoint]** property [WS-Addressing] of the Resolve without waiting for a timer to elapse.
- A Discovery Proxy MAY be configured to reduce multicast traffic on an ad hoc network, in this capacity, if a Discovery Proxy receives a Resolve for other Target Services, it SHOULD respond with a Hello (see Section 4.1.3 Discovery Proxy).

In a managed mode,

- If a Discovery Proxy receives a Resolve request and it has a Target Service that matches the Resolve, it MUST respond with a Resolve Match message. The Resolve Match SHOULD include complete metadata information about the matching Target Service. However, the Resolve Match MAY contain zero matches if the Discovery Proxy has no matching Target Service.

7 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 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:unsignedInt"
                  [SequenceId="xs:anyURI"]?
                   MessageNumber="xs:unsignedInt"
                   ... />
  </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. MUST be compared per RFC 3986 Section 6.2.1 Simple String Comparison [RFC 3986].

/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.

8 Security

8.1 Security Model

This specification does not require that endpoints participating in the discovery process be secure. However, this specification RECOMMENDS that security be used to mitigate various types of attacks (see Section 8.3 Security Considerations).

If a Target Service wishes to secure Hello, Bye, Probe Match and/or Resolve Match, it SHOULD use the compact signature format defined in Section 8.2 Compact Signature Format. A Client MAY choose to ignore Hello, Bye, Probe Match, and/or Resolve Match if it cannot verify the signature.

If a Client wishes to secure Probe and Resolve, it SHOULD use the compact signature format defined in Section 8.2 Compact Signature Format. A Target Service MAY chose to ignore received Probe and/or Resolve if it cannot verify the signature.

There is no requirement for a Target Service to respond to a Probe (or Resolve) if any of the following are true:

- The Target Service is in a different administrative domain than the Client, and the Probe (or Resolve) was sent as multicast, or
- The Target Service fails to verify the signature contained in the Probe (or Resolve).

To avoid participating in a Distributed Denial of Service attack, a Target Service or Discovery Proxy SHOULD NOT respond to a message without a valid signature and MUST NOT respond to a message without a valid signature if the **[reply endpoint]** is not "http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous".

A Client MAY discard a Probe Match (or Resolve Match) if any of the following are true:

- The Probe Match (or Resolve Match) is received MATCH_TIMEOUT seconds or more later than the last corresponding Probe was sent, or
- The Client fails to verify the signature contained in the Probe Match (or Resolve Match).

Table 12 specifies the default value for the MATCH_TIMEOUT parameter.

Table 12: Default value for an application-level parameter.

Parameter	Default Value
MATCH_TIMEOUT	APP_MAX_DELAY + 100 milliseconds

If a Target Service has multiple credentials, it SHOULD send separate Hello, Bye, Probe Match, and/or Resolve Match using different credentials to sign each.

The same security requirements as defined for a Target Service apply to a Discovery Proxy.

8.2 Compact Signature Format

This section defines the signature format for signing UDP unicast and multicast messages.

To minimize the number of XML namespace declarations in messages, the following global attribute is defined:

@d:Id

An alternate ID reference mechanism with the same meaning as @wsu:Id [WS-Security].

This attribute MAY be used to identify which message parts are signed by the compact signature.

The compact signature itself is of the following form:

```
<d:Security ... >
  [<d:Sig Scheme="xs:anyURI"
         [KeyId="xs:base64Binary"]?
          Refs="..."
          Sig="xs:base64Binary"
          ... />]?
  ...
</d:Security>
```

d:Security

A sub-class of the wsse:Security header block [WS-Security] that has the same processing model and rules but is restricted in terms of content and usage. The d:Sig child element provides a compact message signature. Its format is a compact form of XML Signature. To process the signature, the compact form is parsed, and an XML Signature ds:SignedInfo block is created and used for signature verification.

d:Security/@s11:mustUnderstand | d:Security/@s12:mustUnderstand

Processing of the d:Security header block is not mandatory; therefore, the d:Security header block SHOULD NOT be marked mustUnderstand with a value of "true".

d:Security/d:Sig/@Scheme

The governing scheme of the signature. Provides exactly one algorithm for digests and signatures.

The value MUST be compared per RFC 3986 Section 6.2.1 Simple String Comparison [RFC 3986].

d:Security/d:Sig/@Scheme = " http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/rsa"

Exclusive C14N is used for all canonicalization, SHA1 is used for all digests, and Signatures use RSA. Specifically:

http://www.w3.org/2001/10/xml-exc-c14n#

http://www.w3.org/2000/09/xmldsig#sha1

http://www.w3.org/2000/09/xmldsig#rsa-sha1

d:Security/d:Sig/@KeyId

The key identifier of the signing token. MUST be specified if a public key token is used. If included, MUST be Subject Key Identifier (see [RFC 5380] Section 4.2.1.2) of the signing token. If the signing token does not have a Subject Key Identifier, it MUST be the SHA-1 hash of the public key of the signing token. If omitted, the semantics are undefined.

d:Security/d:Sig/@Refs

Parts of the message that have been canonicalized and digested. Each part is referenced by @d:Id (see above). Only immediate children of the security header, top-level SOAP header blocks (/s:Envelope/s:Header/*), and the full SOAP Body (/s:Envelope/s:Body) can be referenced in this list. The value is a space-separated list of IDs to elements within the message.

d:Security/d:Sig/@Sig

The value of the signature.

Table 13 lists an example compact signature.

Table 13: Example compact signature.

```
<d:Sig xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09"
        Scheme="http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09/rsa"
        KeyId="Dx42/9g="
        Refs="ID1"
        Sig="ru5Ef76xGz5Y5IB2iAzDuMvR5Tg=" />

```

A compact signature is expanded into an XML Signature `ds:SignedInfo` using the following pseudo-code.

1. Create an XML Signature `ds:SignedInfo` block. Because canonicalization includes the namespace prefix, this MUST use an XML namespace prefix of "`ds`" so each party can compute a consistent digest value.
2. Populate the block with the appropriate canonicalization and algorithm blocks based on the scheme in `d:Security/d:Sig/@Scheme`.
    - First add a `ds:CanonicalizationMethod` element.
    - Next add a `ds:SignatureMethod` element.
3. For each ID in `d:Security/d:Sig/@Refs` create a corresponding XML Signature Reference element to the identified part (using URI fragments) annotated with the canonicalization and digest algorithms from the scheme in `d:Security/d:Sig/@Scheme`. Note that individual digests need to be computed on the fly.
    - Add a `ds:Reference` element.
    - The `@URI` attribute's value is a "`#`" followed by the specified ID.
    - Inside the `ds:Reference` element add a `ds:Transforms` element that contains a `ds:Transform` element indicating the selected canonicalization algorithm.
    - Inside the `ds:Reference` element add a `ds:DigestMethod` element.
    - Inside the `ds:Reference` element add a `ds:DigestValue` element.
4. Compute the final signature, and verify that it matches.
5. `d:Security/d:Sig/@KeyId`, if present, can be processed as a `SecurityTokenReference` [WS-Security] with an embedded `KeyIdentifier` [WS-Security] specifying the indicated value. While it isn't required to construct a `wsse:SecurityTokenReference` element, the following steps illustrate how one would be created:
6. Create a `wsse:SecurityTokenReference` element.
7. Within this, add a `wsse:KeyIdentifier` element with the value of the `KeyId` attribute's value.

Table 14 lists the expanded form corresponding to the compact form in Table 13.

Table 14: Example expanded signature.

```
<ds:Signature
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" >
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
        Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <ds:SignatureMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <ds:Reference URI="#ID1" >
      <ds:Transforms>
        <ds:Transform
            Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      </ds:Transforms>
      <ds:DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <ds:DigestValue>ODE3NDkyNzI5</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>
    ru5Ef76xGz5Y5IB2iAzDuMvR5Tg=
  </ds:SignatureValue>
  <ds:KeyInfo>
    <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier>Dx42/9g=</wsse:KeyIdentifier>
    </wsse:SecurityTokenReference>
  </ds:KeyInfo>
</ds:Signature>

```

8.3 Security Considerations

Message discovery, both announcements and searches, are subject to a wide variety of attacks. Therefore communication should be secured using the mechanisms described in Section 8.2 Compact Signature Format.

The following list summarizes common classes of attacks and mitigations provided by this protocol:

- **Message alteration** – Message content may be changed by an attacker. To prevent this, the message should be signed. The Body and all relevant headers should be included in the signature. Specifically, the WS-Addressing [WS-Addressing] headers and any headers identified in Endpoint References should be signed together with the Body to "bind" them together.
- **Availability (Denial of Service)** – An attacker may send messages that consume resources. To prevent this, a signature assures that a message is of genuine origin. To avoid unnecessary processing, the signature should be validated before performing beginning any significant processing of message content.
- **Replay** – An attacker may resend a valid message and cause duplicate processing. To prevent this, a replayed message is detected by a duplicate **[message id]** property [WS-Addressing] and should be discarded.
- **Spoofing** – An attacker sends a message that pretends to be of genuine origin. To prevent this, the signature should be unique to the sender.

To provide mitigation against other possible attacks, e.g., message disclosure, mechanisms defined in WS-Security [WS-Security], WS-SecureConversation [WS-SecureConversation], and/or WS-Trust [WS-Trust] may be applied.

If a Client communicates with a Discovery Proxy, the Client should establish end-to-end security with the Discovery Proxy; to improve the efficiency of security operations, the Client should establish a security context using the mechanisms described in WS-Trust [WS-Trust] and WS-SecureConversation [WS-SecureConversation]. In such cases, separate derived keys should be used to secure each message.

9 Conformance

An endpoint MAY implement more than one of the roles; Target Service, Discovery Proxy, and Client; and MAY implement it in more than one of the modes; ad hoc and managed; however, for each implemented role and mode, 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 and modes it implements.

Normative text within this specification takes precedence over normative outlines, which in turn take precedence over the XML Schema [XML Schema Part 1, Part 2] and WSDL [WSDL 1.1] descriptions, which in turn take precedence over examples.

A. Acknowledgements

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

Participants:

Geoff Bullen, Microsoft Corporation
Steve Carter, Novell
Dan Conti, Microsoft Corporation
Doug Davis, IBM
Scott deDeugd, IBM
Dan Driscoll, Microsoft Corporation
Colleen Evans, Microsoft Corporation
Max Feingold, Microsoft Corporation
Travis Grigsby, IBM
Francois Jammes, Schneider Electric
Ram Jeyaraman, Microsoft Corporation
Mike Kaiser, IBM
Supun Kamburugamuva, WSO2
Devon Kemp, Canon Inc.
Akira Kishida, Canon Inc.
Mark Little, Red Hat
Dr. Ingo Lueck, Technische Universitaet Dortmund
Jonathan Marsh, WSO2
Carl Mattocks
Antoine Mensch
Jaime Meritt, Progress Software
Vipul Modi, Microsoft Corporation
Anthony Nadalin, IBM
Tadahiro Nakamura, Canon Inc.
Masahiro Nishio, Canon Inc.
Toby Nixon, Microsoft Corporation
Shin Ohtake, Fuji Xerox Co., Ltd.
Venkat Reddy, CA
Alain Regnier, Ricoh Company, Ltd.
Hitoshi Sekine, Ricoh Company, Ltd.
Hiroshi Tamura, Ricoh Company, Ltd.
Minoru Torii, Canon Inc.
Asir S Vedamuthu, Microsoft Corporation
David Whitehead, Lexmark International Inc.
Don Wright, Lexmark International Inc.
Prasad Yendluri, Software AG, Inc.
Elmar Zeeb, University of Rostock
Gottfried Zimmermann

Co-Developers of the initial contributions:

This document is based on initial contributions to the OASIS WS-DD Technical Committee by the following co-developers.

Gopal Kakivaya, Microsoft Corporation
Devon Kemp, Canon Inc.
Thomas Kuehnel, Microsoft Corporation
Brad Lovering, Microsoft Corporation
Bryan Roe, Intel

Christopher St. John, Software AG
Jeffrey Schlimmer (Editor), Microsoft Corporation
Guillaume Simonnet, Microsoft Corporation
Doug Walter, Microsoft Corporation
Jack Weast, Intel
Yevgeniy Yarmosh, Intel
Prasad Yendluri, Software AG

Acknowledgements of the initial contributions:

The following individuals have provided invaluable input to the original contributions and were acknowledged in the initial contributions.

Don Box, Microsoft Corporation
Shannon Chan, Microsoft Corporation
Dan Conti, Microsoft Corporation
Ken Cooper, Microsoft Corporation
Mike Fenelon, Microsoft Corporation
Omri Gazitt Microsoft Corporation,
Bertus Greeff, Microsoft Corporation
Rob Hain, Microsoft Corporation
Richard Hasha, Microsoft Corporation
Erin Honeycutt, Microsoft Corporation
Christian Huitema, Microsoft Corporation
Chris Kaler, Microsoft Corporation
Umesh Madan, Microsoft Corporation
Vipul Modi, Microsoft Corporation
Jeff Parham, Microsoft Corporation
Yaniv Pessach, Microsoft Corporation
Stefan Pharies, Microsoft Corporation
Dale Sather, Microsoft Corporation
Matt Tavis, Microsoft Corporation

1490 B. Revision History

1491 [optional; should not be included in OASIS Standards]

Revision	Date	Editor	Changes Made
wd-01	09/16/2008	Vipul Modi	Created the initial working draft by converting the input specification to OASIS template.
wd-01	09/16/2008	Vipul Modi	Authoritative format changed to docx from doc
wd-01	09/19/2008	Vipul Modi	Adjusted the location of the document as per the format decided on 09/18/2008 during F2F meeting day 3.
wd-01	09/24/2008	Vipul Modi	Fixed broken links for cross referencing Table, Figure and Section.
wd-02	09/26/2008	Vipul Modi	Incorporated proposals for the following issues: 018 - Move Application Sequencing from Appendix to main specification 019 - Combine security section under a single top level heading 020 - XSD and WSDL files as separate resources 047 - Replace reference to RFC 2396 with RFC 3986 048 - Probe requirement in ResolveMatch section 050 - The UUIDs URIs do not use UUID URN Namespace defined by RFC 4122 054 -Remove support for SOAP 1.1 058 - Remove transport specification retransmission notes in ProbeMatch and ResolveMatch sections 059 - Follow WSDL naming conventions in naming messages and part names 062 - Description of Scopes element for Probe does not mention that whitespace is not allowed 063 -Clarify matching behavior for empty <d:Types>, <d:Scopes> element 064 -Clarify matching algorithm for @MatchBy, @Scheme and @SequenceId 065 - Terminologies should not make normative text like statements 066 - RelationshipType attribute is not required 067- Define KeyID content in the d:Sig 061- Use OASIS assigned namespace
wd-03	10/20/2008	Vipul Modi	Incorporated the proposal for the following issues 022 - request-response MEP for communicating with proxy 034 - Discovery proxy and multicast suppression requirement

			035 - define protocol assigment/binding for managed mode 036 - discovery messages and managed mode 049 - forced managed mode transition for the client
cd-01	10/21/2008	Vipul Modi	Created first committee draft by from working draft 03. Removed all change bars.
cd-01	1/27/2009	Vipul Modi	Changes to comply with the OASIS document format. * Namespace changed from http://docs.oasis-open.org/ws-dd/discovery/2008/09 to http://docs.oasis-open.org/ws-dd/ns/discovery/2008/09 * Created a Section 1.5 Terminology. Section 2.1 Terminology moved under it and renamed as 1.5.2 Terms and Definitions. * Section 2.2 is moved under Section 1.5 and became Section 1.5.1. * Section 2.3 is now Section 1.6 * Section 2.4 is now Section 1.7 * Section 2.5 is now Section 3 * Section 2.6 Compliance is now Section 9 Conformance. * Cover Page: Previous Version is marked as N/A. * Cover Page: Latest Approved Version is removed. * Cover Page: Corrected errors in the hyperlinks * Added the names of the TC members in the acknowledgement section.