Understanding Devices Profile for Web Services, Web Services Discovery, and SOAP-over-UDP

September 10, 2008

Version 1.0

Authors
Ram Jeyaraman, Microsoft Corporation (Editor)
Vipul Modi, Microsoft Corporation
Dan Driscoll, Microsoft Corporation
Geoff Bullen, Microsoft Corporation
Toby Nixon, Microsoft Corporation

Copyright Notice

Summary

The primary goal of this document is to increase understanding of the use of Devices Profile for Web Services (DPWS) **[DPWS]**, Web Services Discovery (WS-Discovery) **[WS-DISCOVERY]**, and SOAP-over-UDP [SOAP-over-UDP] specifications using relevant usage scenarios to illustrate and motivate the use of those specifications. Refer to the respective specifications for complete normative description and semantics of the concerned protocols. This document also provides implementation guidance on the use of WS-Discovery with multi-homed devices and systems.

The major sections in this document are:

- Section 2 – Usage scenario: Printing a document – illustrates the use of DPWS to discover, connect to, and communicate with a device on the network. This scenario assumes an ad hoc network where no specialized networking services or intermediaries are present.
- Section 3 – Usage scenario: Service discovery using a proxy – illustrates the managed discovery mechanism. This scenario assumes the availability of a specialized network service, in this case, an enterprise repository acting as a discovery proxy.
- Section 4 – Implementation guidance – provides implementation guidance on the use of WS-Discovery with multi-homed devices and systems.

This is a non-normative document. The information contained in this document is intended for a technical audience such as developers and implementers who seek a deeper technical understanding of the use of DPWS, WS-Discovery and SOAP-over-UDP specifications. This document does not cover secure usage scenarios.

Contents

1 Introduction 3
1.1 Devices and Services 3
1.2 DPWS 4
1.3 WS-Discovery 4
1.4 SOAP-over-UDP 5
2 Usage scenario: Printing a document 5
2.1 Printer WSDL (sample) 7
2.2 Detailed description 9
3 Usage scenario: Service discovery using a proxy 22
3.1 Detailed description 24
4 Implementation guidance on using WS-Discovery with multi-homed devices and systems 29
4.1 Multi-homed devices and systems 29

4.2 Implementation guidance 31
4.2.1 General 31
4.2.2 Hello and Bye 31
4.2.3 Probe and Resolve 32
5 Conclusion 32
Appendix A. Acknowledgements 32
Appendix B. XML namespaces 32
Appendix C. References 33

1 Introduction

1.1 Devices and Services

From the point of view of DPWS, WS-Discovery and SOAP-over-UDP specifications, a *service* is a Web Service. DPWS defines a special service, called a *device*, as a Web Service, whose function is to participate in discovery, and to describe other services available or hosted in the same physical device container (see Figure 1). The type for this hosting service is wsdp:Device (see appendix section XML namespaces). Services that are not a DPWS device may also participate in WS-Discovery.

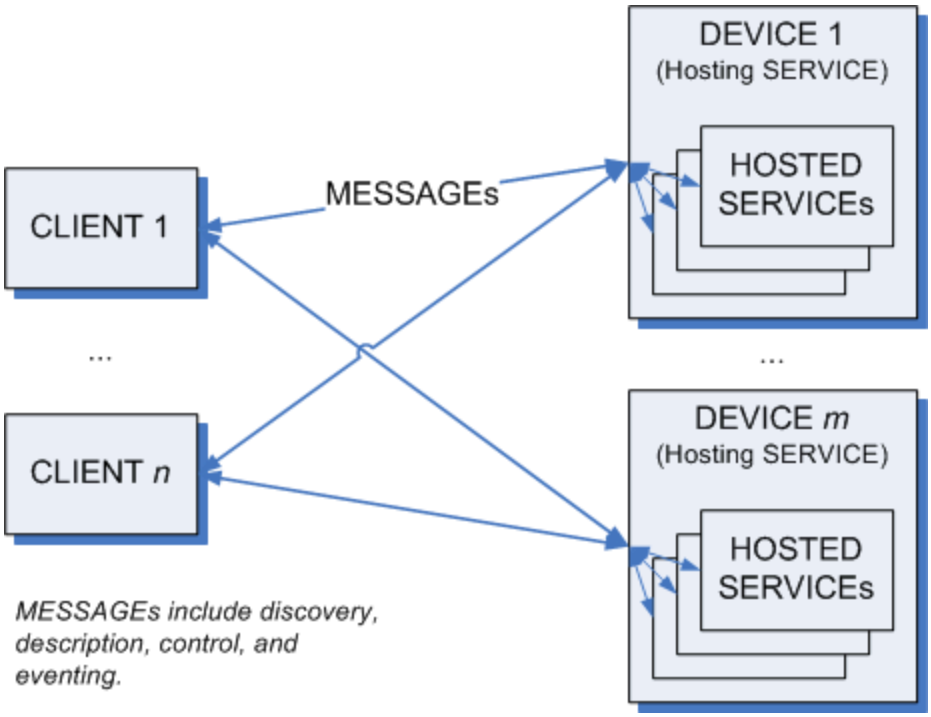

Figure 1. Devices and Services

1.2 DPWS

DPWS is a profile of Web Services protocols consisting of a minimal set of implementation constraints to enable secure Web Service messaging, discovery, description, and eventing on resource-constrained endpoints. It defines an extensible metadata model for describing the characteristics of devices. It defines the metadata format that allows services of wsdp:Device type to describe their hosted services, such as printer services and scanner services. It defines a policy assertion that allows devices to indicate compliance with this profile. It provides guidance on security.

DPWS uses SOAP **[SOAP]**, WS-Addressing **[WS-Addressing]**, and MTOM/XOP **[MTOM]** for messaging. It supports SOAP-over-HTTP **[SOAP-over-HTTP]** and SOAP-over-UDP bindings. It uses WS-Discovery for discovering a device (hosting service), WS-Eventing **[WS-Eventing]** for optionally setting up and managing subscriptions to events from the device. It uses Web Services Description Language **[WSDL]** to describe the device, Web Services Metadata Exchange **[WS-MEX]** to define metadata about the device, Web Services Policy **[WS-Policy]** to define a policy assertion to indicate compliance of the device with DPWS, and WS-Transfer **[WS-Transfer]** to retrieve service description and metadata information about the device.

1.3 WS-Discovery

WS-Discovery **[WS-DISCOVERY]**provides a Web Services based lightweight dynamic discovery protocol to locate Web Services. It does not require specialized network intermediaries to aid discovery. It is transport independent; it may be used over HTTP **[HTTP]**, UDP, or other transports. It defines a compact signature format for cryptographically signing WS-Discovery protocol messages sent over UDP transport.

WS-Discovery allows discovery of services in ad hoc networks with a minimum of networking services (e.g., no DNS or directory services). It leverages, when available, network services (e.g., DNS or directory services) to reduce network traffic in managed networks, where such services exist. It allows for smooth transitions between ad hoc and managed networks.

1.4 SOAP-over-UDP

SOAP-over-UDP is a binding of SOAP to UDP (User Datagram Protocol) **[UDP]** transport, including message patterns (one-way, request-response, and multicast), message encoding, URI scheme, security considerations, message re-transmission behavior, and duplicate detection mechanisms. This binding describes how to transmit a SOAP message inside a UDP packet, including guidance on how to secure the SOAP message in spite of the UDP packet size limitation (64k).

2 Usage scenario: Printing a document

This scenario illustrates the use of DPWS to discover, connect to, and communicate with a device on the network. This scenario assumes an ad hoc network where no specialized networking services or intermediaries are present. This scenario describes the use of the ad hoc discovery mechanism to discover a device on the local network, the use of the SOAP-over-UDP binding to multicast discovery messages, the use of **[WS-MEX]** and **[WS-Transfer]** to obtain metadata information about the device, the use of **[SOAP]** and **[MTOM]** to send and receive messages from the device, and the use of **[WS-Eventing]** to subscribe to event notifications from the device. This scenario uses a sample **[WSDL]** for the device to describe the message exchanges. The sample WSDL and printer-specific messages in this example are supplied solely to illustrate the use of DPWS.

In this scenario, a human user with a laptop walks into a location that has PCs, Servers, printers, scanners, and other devices all connected to a local network (see Figure 2). This scenario steps through the various message exchanges that occur between the laptop and the devices on the network, as the laptop enters the network and as the user attempts to print a document stored in the laptop.

Figure 2. Devices on a network

A high-level description of this scenario:

1. The user with the laptop joins the network. The user attempts to print a document created by an application in the laptop. The laptop attempts to locate printers on the network based on a search criteria specified by the application. This is done using WS-Discovery.
2. The laptop communicates with each of the matching printers to gather metadata information about them. The application displays the list of available printers that match the search criteria. The user does not find the printer that she/he is looking for. The user walks over to a network printer that is turned off and manually turns it on. The printer turns on and announces itself on the network. This is done using WS-Discovery.
3. The announcement is received by the laptop. The laptop requests metadata from the printer to get more information such as manufacturer name, model name, firmware version, WSDL description, etc. This is done using WS-MetadataExchange **[WS-MEX]** and WS-Transfer **[WS-Transfer]**. The laptop uses the metadata about the printer to present a localized name in the user interface (UI) displayed (to the user), identify any print services on the device (printer), etc.
4. The laptop displays the newly discovered printer by adding an icon (representing the new printer) to the list of printers displayed in the UI.
5. The user manually selects this printer from the list of available printers. The laptop subscribes to events from the printer such as print job status, etc. This is done using WS-Eventing **[WS-Eventing]**.
6. The laptop sends the document to the printer.
7. The printer sends events to the laptop notifying the progress and completion of the print job.
8. The user decides to shut down the laptop. The laptop unsubscribes itself from receiving event notifications from the printer. This is done using WS-Eventing.
9. The user turns off the printer. The printer sends a notification announcing its departure from the network. This is done using WS-Discovery.

The following is a more detailed description of this scenario. It contains a sample WSDL for a print device, and sample message exchanges between the laptop and the devices on the network, including the printer.

2.1 Printer WSDL (sample)

WSDL descriptions for specific device classes such as printers, scanners and conference room projectors are defined outside the DPWS specification. The following is a sample WSDL for a print device. This sample WSDL describes the operations supported by the print device used in this scenario.

Sample Print Device WSDL:

```
<wsdl:definitions
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdp="http://schemas.xmlsoap.org/ws/2006/02/devprof"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wsu=
        "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    targetNamespace="http://example.com/wsdp/sample/print">

    <wsp:Policy wsu:Id="DevicePolicy">
        <wsdp:Profile/>
    </wsp:Policy>

    <wsdl:portType name="PrintDeviceType"/>

</wsdl:definitions>
```

Sample Printer Service WSDL:

```
<definitions
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsoap12=" http://www.w3.org/2003/05/soap-envelope"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:wsdp="http://schemas.xmlsoap.org/ws/2006/02/devprof"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsu=
        "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    targetNamespace="http://example.com/wsdp/sample/print"
    name="SamplePrinterService">

    <wsp:Policy wsu:Id="ServicePolicy">
        <wsdp:Profile />
    </wsp:Policy>

    <types>
        <xs:schema targetNamespace="http://schemas example.com/wsdp/sample/print">
            <xs:include schemaLocation="SamplePrint.xsd"/>
        </xs:schema>
    </types>

    <message name="CreatePrintJobRequestMsg">
        <part name="body" element="..."/>
    </message>
```

```
    <message name="CreatePrintJobResponseMsg">
        <part name="body" element="..."/>
    </message>
    <message name="SendDocumentRequestMsg">
        <part name="body" element="..."/>
    </message>
    <message name="SendDocumentResponseMsg">
        <part name="body" element="..."/>
    </message>
    <message name="JobStatusEventMsg">
        <part name="body" element="..."/>
    </message>
    <message name="JobEndStateEventMsg">
        <part name="body" element="..."/>
    </message>

    <portType name="PrinterServiceType" wse:EventSource="true">
        <operation name="CreatePrintJob">
            <input message="spt:CreatePrintJobRequestMsg"
                   wsa:Action="http://example.com/wsdp/sample/print/CreatePrintJob"/>
            <output message="spt:CreatePrintJobResponseMsg"
                   wsa:Action="http://example.com/wsdp/sample/print/CreatePrintJobResponse"/>
        </operation>
        <operation name="SendDocument">
            <input message="spt:SendDocumentRequestMsg"
                   wsa:Action="http://example.com/wsdp/sample/print/SendDocument"/>
            <output message="spt:SendDocumentResponseMsg"
                   wsa:Action="http://example.com/wsdp/sample/print/SendDocumentResponse"/>
        </operation>
        <operation name="JobStatusEvent">
            <output message="spt:JobStatusEventMsg"
                   wsa:Action="http://example.com/wsdp/sample/print/JobStatusEvent"/>
        </operation>
        <operation name="JobEndStateEvent">
            <output message="spt:JobEndStateEventMsg"
                   wsa:Action="http://example.com/wsdp/sample/print/JobEndStateEvent"/>
        </operation>
    </portType>

    <binding name="PrinterServiceBinding" type="...">
        <wsoap12:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsp:PolicyReference URI="#ServicePolicy"/>
        <operation name="CreatePrintJob">
            <wsoap12:operation
                soapAction="http://example.com/wsdp/sample/print/CreatePrintJob"
                soapActionRequired="true"/>
            <input>
                <wsoap12:body use="literal"/>
            </input>
            <output>
                <wsoap12:body use="literal"/>
            </output>
        </operation>
        <operation name="SendDocument">
            <wsoap12:operation
                soapAction="http://example.com/wsdp/sample/print/SendDocument"
                soapActionRequired="true" />
            <input>
                <wsoap12:body use="literal" />
            </input>
            <output>
```

```
                <wsoap12:body use="literal" />
            </output>
        </operation>
        <operation name="JobStatusEvent">
            <wsoap12:operation
                soapAction="http://example.com/wsdp/sample/print/JobStatusEvent"
                soapActionRequired="true" />
            <output>
                <wsoap12:body use="literal" />
            </output>
        </operation>
        <operation name="JobEndStateEvent">
            <wsoap12:operation
                soapAction="http://example.com/wsdp/sample/print/JobEndStateEvent"
                soapActionRequired="true" />
            <output>
                <wsoap12:body use="literal" />
            </output>
        </operation>
    </binding>
</definitions>
```

2.2 Detailed description

The user with the laptop joins the network. The user attempts to print a document created by an application in the laptop. The laptop attempts to locate printers on the network based on a search criteria specified by the application. This is done by sending a UDP multicast message (using the SOAP-over-UDP binding) carrying a SOAP envelope that contains a WS-Discovery Probe message. The Probe message contains the search criteria, in this case, the type of the printer. Print devices on the local subnet that match the search criteria specified in the Probe message will respond with a unicast WS-Discovery Probe Match message (sent using the SOAP-over-UDP binding); each Probe Match message contains network location information about the specific printer.

```
<!-- WS-Discovery multicast Probe message -->

<!--
SOAP-over-UDP message sent multicast from 192.168.0.200:65938 to 239.255.255.250:3702
-->

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery"
    xmlns:spt="http://example.com/wsdp/sample/print">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:ac9823ee-6813-498c-8c1b-6272a22353f7</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/Probe
        </wsa:Action>
        <wsa:To>urn:schemas-xmlsoap-org:ws:2005:04:discovery</wsa:To>
    </wsoap12:Header>
    <wsoap12:Body>
        <wsdisco:Probe>
            <wsdisco:Types>wsdp:Device spt:PrintDeviceType</wsdisco:Types>
        </wsdisco:Probe>
```

```
    </wsoap12:Body>
</wsoap12:Envelope>
```

```
<!-- WS-Discovery unicast Probe Match message -->

<!--
SOAP-over-UDP message sent unicast from 192.168.0.105:3702 to 192.168.0.200:65938
-->

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery"
    xmlns:wsdp=”http://schemas.xmlsoap.org/ws/2006/02/devprof”
    xmlns:spt=" http://example.com/wsdp/sample/print ">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:0a6dc791-2be6-4991-9af1-454778a1917a</wsa:MessageID>
        <wsa:RelatesTo>urn:uuid:ac9823ee-6813-498c-8c1b-6272a22353f7</wsa:RelatedTo>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/ProbeMatches
        </wsa:Action>
        <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
        <wsdisco:AppSequence InstanceId="7094203799" MessageNumber="1"/>
    </wsoap12:Header>
    <soap12:Body>
        <wsdisco:ProbeMatches>
            <wsdisco:ProbeMatch>
                <wsa:EndpointReference>
                    <wsa:Address>urn:uuid:e20000d1-ca45-5fee-a376-112233445555</wsa:Address>
                </wsa:EndpointReference>
                <wsdisco:Types>wsdp:Device spt:PrintDeviceType</wsdisco:Types>
                <wsdisco:XAddrs>http://192.168.0.105:80/TestPrintDevice</wsdisco:XAddrs>
                <wsdisco:MetadataVersion>1</wsdisco:MetadataVersion>
            </wsdisco:ProbeMatch>
        </wsdisco:ProbeMatches>
    </wsoap12:Body>
</wsoap12:Envelope>
```

The laptop communicates with each of the matching printers to gather metadata information about them. The application displays the list of available printers that match the search criteria. The user does not find the printer that she/he is looking for. The user walks over to a network printer that is turned off and manually turns it on. The printer turns on and announces itself on the network by sending a UDP multicast message (using the SOAP-over-UDP binding) carrying a SOAP envelope that contains a WS-Discovery Hello message.

```
<!-- WS-Discovery multicast Hello message from printer -->

<!--
SOAP-over-UDP message sent multicast from 192.168.0.101:65938 to 239.255.255.250:3702
-->

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
```

```
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery"
    xmlns:wsdp="http://schemas.xmlsoap.org/ws/2006/02/devprof"
    xmlns:spt="http://example.com/wsdp/sample/print">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:ac8523ee-6813-498c-8c1b-6272a22353f7</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/Hello
        </wsa:Action>
        <wsa:To>urn:schemas-xmlsoap-org:ws:2005:04:discovery</wsa:To>
        <wsdisco:AppSequence InstanceId="5094203799" MessageNumber="1"/>
    </wsoap12:Header>
    <wsoap12:Body>
        <wsdisco:Hello>
            <wsa:EndpointReference>
                <wsa:Address>urn:uuid:d10000d1-ca45-5fee-a376-112233445555</wsa:Address>
            </wsa:EndpointReference>
            <wsdisco:Types>wsdp:Device spt:PrintDeviceType</wsdisco:Types>
            <wsdisco:MetadataVersion>1</wsdisco:MetadataVersion>
        </wsdisco:Hello>
    </wsoap12:Body>
</ wsoap12:Envelope>
```

The announcement (Hello message) is received by the laptop. In this case, the Hello message contains the endpoint address but does not contain a transport address. In order to display information about the newly added printer to the user, the laptop needs to contact the printer to obtain metadata information about it. For this, the laptop needs the transport address of the printer. The laptop sends a multicast Resolve message (using the SOAP-over-UDP binding) to resolve the endpoint address (received via the Hello message) to a transport address. The newly added printer receives the multicast Resolve message and responds with a unicast Resolve Match message containing its transport address.

```
<!-- WS-Discovery multicast Resolve message -->

<!--
SOAP-over-UDP message sent multicast from 192.168.0.200:65938 to 239.255.255.250:3702
-->

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:xy8923ee-6813-498c-8c1b-6272a22353f7</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/Resolve
        </wsa:Action>
        <wsa:To>urn:schemas-xmlsoap-org:ws:2005:04:discovery</wsa:To>
    </wsoap12:Header>
    <wsoap12:Body>
        <wsdisco:Resolve>
            <wsa:EndpointReference>
                <wsa:Address>urn:uuid:d10000d1-ca45-5fee-a376-112233445555</wsa:Address>
            </wsa:EndpointReference>
        </wsdisco:Resolve>
    </wsoap12:Body>
</wsoap12:Envelope>
```

```
<!-- WS-Discovery unicast Resolve Match message -->

<!--
SOAP-over-UDP message sent unicast from 192.168.0.101:3702 to 192.168.0.200:65938
-->


<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery"
    xmlns:wsdp=”http://schemas.xmlsoap.org/ws/2006/02/devprof”
    xmlns:spt=" http://example.com/wsdp/sample/print ">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:1b7dc791-2be6-4971-9af1-954778a1917a</wsa:MessageID>
        <wsa:RelatesTo>urn:uuid:xy8923ee-6813-498c-8c1b-6272a22353f7</wsa:RelatedTo>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/ResolveMatches
        </wsa:Action>
        <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
        <wsdisco:AppSequence InstanceId="5094203799" MessageNumber="2"/>
    </wsoap12:Header>
    <soap12:Body>
        <wsdisco:ResolveMatches>
            <wsdisco:ResolveMatch>
                <wsa:EndpointReference>
                    <wsa:Address>urn:uuid:d10000d1-ca45-5fee-a376-112233445555</wsa:Address>
                </wsa:EndpointReference>
                <wsdisco:Types>wsdp:Device spt:PrintDeviceType</wsdisco:Types>
                <wsdisco:XAddrs>http://192.168.0.101:80/SamplePrintDevice</wsdisco:XAddrs>
                <wsdisco:MetadataVersion>1</wsdisco:MetadataVersion>
            </wsdisco:ResolveMatch>
        </wsdisco:ProbeMatches>
    </wsoap12:Body>
</wsoap12:Envelope>
```

The laptop uses the transport address to request metadata from the printer to get more information such as manufacturer name, model name, firmware version, WSDL description, etc. This is done by sending a SOAP envelope containing a WS-Transfer Get message (sent using SOAP-over-HTTP **[SOAP-over-HTTP]** binding) to the chosen printer. The printer sends a WS-Transfer GetResponse (sent using the SOAP-over-HTTP binding) containing the metadata. This message is sent to the transport address obtained via the Resolve Match message. The laptop uses the metadata about the printer to present a localized name in the user interface (UI) displayed (to the user), identify any print services on the device (printer), etc.

```
<!-- WS-Transfer Get message -->

POST /SamplePrintDevice
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.101:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
```

```
Pragma: no-cache

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:2a9dc791-2be6-4991-9af1-454778a1917a</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
        </wsa:Action>
        <wsa:To>urn:uuid:d10000d1-ca45-5fee-a376-112233445555</wsa:To>
        <wsa:ReplyTo>
            http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
        </wsa:ReplyTo>
    </wsoap12:Header>
    <wsoap12:Body/>
</wsoap12:Envelope>
```

```
<!-- WS-Transfer GetResponse message -->

HTTP/1.1 200
Content-type: application/soap+xml
Connection: close
Content-length: ...

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsdp="http://schemas.xmlsoap.org/ws/2006/02/devprof"
    xmlns:mex="http://schemas.xmlsoap.org/ws/2004/09/mex"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:718dc796-2be6-4991-9af1-454778a1917a</wsa:MessageID>
        <wsa:RelatesTo>urn:uuid:2a9dc791-2be6-4991-9af1-454778a1917a</wsa:RelatesTo>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
        </wsa:Action>
        <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
    </wsoap12:Header>
    <wsoap12:Body>
        <mex:Metadata>
            <mex:MetadataSection
                   Dialect="http://schemas.xmlsoap.org/ws/2006/02/devprof/ThisModel">
                <wsdp:ThisModel>
                    <wsdp:Manufacturer>ACME Manufacturing</wsdp:Manufacturer>
                    <wsdp:ModelName xml:lang="en-US">ColorBeam 9</wsdp:ModelName>
                </wsdp:ThisModel>
            </mex:MetadataSection>
            <mex:MetadataSection
                   Dialect="http://schemas.xmlsoap.org/ws/2006/02/devprof/ThisDevice">
                <wsdp:ThisDevice>
                    <wsdp:FriendlyName xml:lang="en-US">
                        ACME ColourBeam Printer
                    </wsdp:FriendlyName>
                </wsdp:ThisDevice>
            </mex:MetadataSection>
            <mex:MetadataSection
                   Dialect="http://schemas.xmlsoap.org/ws/2006/02/devprof/Relationship">
                <wsdp:Relationship Type="http://schemas.xmlsoap.org/ws/2006/02/devprof/host">
```

```
                <wsdp:Hosted>
                    <wsa:EndpointReference>
                        <wsa:Address>
                            http://192.168.0.101:80/SamplePrintService0
                        </wsa:Address>
                    </wsa:EndpointReference>
                    <wsdp:Types
                            xmlns:spt="http://example.com/wsdp/sample/print">
                        spt:PrinterServiceType
                    </wsdp:Types>
                    <wsdp:ServiceId>
                        http://example.com/sample/print/PrintService
                    </wsdp:ServiceId>
                </wsdp:Hosted>
            </wsdp:Relationship>
        </mex:MetadataSection>
        <mex:MetadataSection Dialect="http://schemas.xmlsoap.org/wsdl">
            <mex:MetadataReference>
                <wsa:Address>urn:uuid:d10000d1-ca45-5fee-a376-112233445555</wsa:Address>
                <wsa:ReferenceParameters>
                    <spt:Acme xmlns:spt="http://example.com/wsdp/sample/print">
                        WSDL
                    </spt:Acme>
                </wsa:ReferenceParameters>
            </mex:MetadataReference>
        </mex:MetadataSection>
    </mex:Metadata>
  </wsoap12:Body>
</wsoap12:Envelope>
```

The laptop displays the newly discovered printer by adding an icon (representing the new printer) to the list of printers displayed in the UI. The user manually selects this printer from the list of available printers. The laptop subscribes to events from the printer such as print job status, etc. This is done by sending a SOAP envelope containing a WS-Eventing Subscribe message (sent using the SOAP-over-HTTP binding) to the chosen printer. The printer responds by sending a WS-Eventing SubscribeResponse message via the HTTP response channel.

```
<!-- WS-Eventing Subscribe message -->

POST /SamplePrintService0
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.101:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:7a7dc791-2be6-4996-9af8-454778a1917a</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2004/08/eventing/Subscribe
        </wsa:Action>
```

```
        <wsa:To>http://192.168.0.101:80/SamplePrintService0</wsa:To>
        <wsa:ReplyTo>
            http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
        </wsa:ReplyTo>
    </wsoap12:Header>
    <wsoap12:Body>
        <wse:Subscribe>
            <wse:EndTo>
                <wsa:Address>http://192.168.0.200:80/PrintEventSink</wsa:Address>
                <wsa:ReferenceParameters>
                    <wse:Identifier>2597</wse:Identifier>
                </wsa:ReferenceParameters>
            </wse:EndTo>
            <wse:Delivery>
                <wse:NotifyTo>
                    <wsa:Address>
                        http://192.168.0.200:80/PrintEventSink
                    </wsa:Address>
                    <wsa:ReferenceParameters>
                        <wse:Identifier>2597</wse:Identifier>
                    </wsa:ReferenceParameters>
                </wse:NotifyTo>
            </wse:Delivery>
            <wse:Expires>PT1H</wse:Expires>
        </wse:Subscribe>
    </wsoap12:Body>
</wsoap12:Envelope>
```

```
<!-- WS-Eventing SubscribeResponse message -->

HTTP/1.1 200
Content-type: application/soap+xml
Connection: close
Content-length: ...

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:9b7ef791-2be6-4996-9ae8-954778a1917a</wsa:MessageID>
        <wsa:RelatesTo>urn:uuid:7a7dc791-2be6-4996-9af8-454778a1917a</wsa:RelatesTo>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2004/08/eventing/SubscribeResponse
        </wsa:Action>
        <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
    </wsoap12:Header>
    <wsoap12:Body>
        <wse:SubscribeResponse>
            <wse:SubscriptionManager>
                <wsa:Address>
                    http://192.168.0.101:80/SamplePrintService0
                </wsa:Address>
                <wsa:ReferenceParameters>
                    <wse:Identifier>
                        urn:uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa
                    </wse:Identifier>
                </wsa:ReferenceParameters>
            </wse:SubscriptionManager>
```

```
          <wse:Expires>PT30M</wse:Expires>
       </wse:SubscribeResponse>
   </wsoap12:Body>
</wsoap12:Envelope>
```

The laptop sets up a print job by sending a job request to the printer. The printer responds with a job id. This message exchange uses the SOAP-over-HTTP binding. The format of this message is defined by the sample printer WSDL.

```
<!-- Request to creat print job -->

POST /SamplePrintService0
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.101:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<wsoap12:Envelope
       xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
       xmlns:spt="http://example.com/wsdp/sample/print">
   <wsoap12:Header>
       <wsa:To>http://192.168.0.101:80/SamplePrintService0</wsa:To>
       <wsa:Action>
           http://example.com/wsdp/sample/print/CreatePrintJob
       </wsa:Action>
       <wsa:MessageID>urn:uuid:7c8ef791-1be5-6996-9ae8-956778a1937a</wsa:MessageID>
   </wsoap12:Header>
   <wsoap12:Body>
       <spt:CreatePrintJobRequest>
          <spt:PrintTicket>
              <spt:JobDescription>
                  <spt:JobName>Example Job</spt:JobName>
                  <spt:JobOriginatingUserName>Ram</spt:JobOriginatingUserName>
              </spt:JobDescription>
              <spt:JobProcessing>
                  <spt:Copies MustHonor="1">3</spt:Copies>
                  <spt:Priority>50</spt:Priority>
              </spt:JobProcessing>
              <spt:DocumentProcessing>
                  <spt:Orientation>Landscape</spt:Orientation>
                  <spt:PrintQuality>Draft</spt:PrintQuality>
                  <spt:Sides>OneSided</spt:Sides>
              </spt:DocumentProcessing>
          </spt:PrintTicket>
       </spt:CreatePrintJobRequest>
   </wsoap12:Body>
</wsoap12:Envelope>
```

```
<!-- Response for creat print job request -->

HTTP/1.1 200
Content-type: application/soap+xml
Connection: close
Content-length: ...
```

```
<wsoap12:Envelope
       xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
       xmlns:spt="http://example.com/wsdp/sample/print">
   <wsoap12:Header>
       <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
       <wsa:Action>
          http://example.com/wsdp/sample/print/CreatePrintJobResponse
       </wsa:Action>
       <wsa:MessageID>urn:uuid:3c8ef793-1be3-6993-9ae8-956778a1937a</wsa:MessageID>
       <wsa:RelatesTo>urn:uuid:7c8ef791-1be5-6996-9ae8-956778a1937a</wsa:RelatesTo>
   </wsoap12:Header>
   <wsoap12:Body>
       <spt:CreatePrintJobResponse>
          <spt:JobId>1</spt:JobId>
       </spt:CreatePrintJobResponse>
   </wsoap12:Body>
</wsoap12:Envelope>
```

The laptop sends the document to the printer. This is done by sending a SOAP envelope containing the document (binary data) to the printer. The SOAP message is transmitted using XML-binary optimized packaging (XOP) via SOAP MTOM **[MTOM]**. The printer sends back a response acknowledging the receipt. This message exchange uses the SOAP-over-HTTP binding.

```
<!-- SOAP envelope (as a MIME Multipart/Related XOP Package) containing the document (binary)
to be printed -->

POST /SamplePrintService0
HTTP/1.1
Host: 192.168.0.101:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache
Content-Type: Multipart/Related;
   boundary=e1ac1c8d-11be-4313-999f-a8b0857d5360-3c31c1fe-39ef-4c34-8efd-918cf79f6;
   type="application/xop+xml";
   start="<xml@example.org>";
   start-info="application/soap+xml"
MIME-Version: 1.0

--e1ac1c8d-11be-4313-999f-a8b0857d5360-3c31c1fe-39ef-4c34-8efd-918cf79f6
Content-Type: application/xop+xml;type=application/soap+xml;charset=UTF-8
Content-Transfer-Encoding: binary
Content-ID: <soap@soap>

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
       xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
       xmlns:spt="http://example.com/wsdp/sample/print"
       xmlns:xop=”http://www.w3.org/2004/08/xop/include”>
   <wsoap12:Header>
       <wsa:To>http://192.168.0.101:80/SamplePrintService0</wsa:To>
       <wsa:Action>
          http://example.com/wsdp/sample/print/SendDocument
       </wsa:Action>
       <wsa:MessageID>urn:uuid:5c7ef791-5be6-6996-9ae8-954778a1917a</wsa:MessageID>
```

```
    </wsoap12:Header>
    <wsoap12:Body>
        <spt:SendDocumentRequest>
            <spt:JobId>1</spt:JobId>
            <spt:DocumentDescription>
                <spt:DocumentId>1</spt:DocumentId>
                <spt:Compression>None</spt:Compression>
                <spt:Format>application/octet-stream</spt:Format>
                <spt:DocumentName>Example.xml</spt:DocumentName>
            </spt:DocumentDescription>
            <spt:DocumentProcessing>
                <spt:Sides>TwoSidedLongEdge</spt:Sides>
            </spt:DocumentProcessing>
            <spt:DocumentData>
                <xop:Include href=”cid:1c696bd7-005a-48d9-9ee9-9adca11f8892@uuid”/>
            </spt:DocumentData>
        </spt:SendDocumentRequest>
    </wsoap12:Body>
</wsoap12:Envelope>

--e1ac1c8d-11be-4313-999f-a8b0857d5360-3c31c1fe-39ef-4c34-8efd-918cf79f6
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Content-ID: <1c696bd7-005a-48d9-9ee9-9adca11f8892@uuid>

Document PDL Data
--e1ac1c8d-11be-4313-999f-a8b0857d5360-3c31c1fe-39ef-4c34-8efd-918cf79f6--
```

```
<!-- SOAP envelope containing the response to the print request -->

HTTP/1.1 200
Content-type: application/soap+xml
Connection: close
Content-length: ...

<wsoap12:Envelope
       xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
    <wsoap12:Header>
        <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
        <wsa:Action>
            http://example.com/wsdp/sample/print/SendDocumentResponse
        </wsa:Action>
        <wsa:MessageID>urn:uuid:6c8ef791-6be6-6996-9ae8-454778a1917a</wsa:MessageID>
        <wsa:RelatesTo>urn:uuid:5c7ef791-5be6-6996-9ae8-954778a1917a</wsa:RelatesTo>
    </wsoap12:Header>
    <wsoap12:Body>
        <spt:SendDocumentResponse/>
    </wsoap12:Body>
</wsoap12:Envelope>
```

The printer sends an event notification (using the SOAP-over-HTTP binding) to the laptop indicating the status of the print job.

```
<!-- Event notification about the status of the print job -->

POST /PrintEventSink
HTTP/1.1
Content-Type: application/soap+xml
```

```
Host: 192.168.0.200:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<wsoap12:Envelope
       xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
       xmlns:spt="http://example.com/wsdp/sample/print">
   <wsoap12:Header>
       <wsa:To>http://192.168.0.200:80/PrintEventSink</wsa:To>
       <wsa:Action>
           http://example.com/wsdp/sample/print/JobStatusEvent
       </wsa:Action>
       <wsa:MessageID>urn:uuid:7a1ef791-6be6-6996-7ae8-654778a1917a</wsa:MessageID>
       <wse:Identifier>2597</wse:Identifier>
   </wsoap12:Header>
   <wsoap12:Body>
       <spt:JobStatusEvent>
           <spt:JobStatus>
               <spt:JobId>1</spt:JobId>
               <spt:JobState>Processing</spt:JobState>
               <spt:JobStateReasons>
                   <spt:JobStateReason>JobSpooling</spt:JobStateReason>
                   <spt:JobStateReason>JobPrinting</spt:JobStateReason>
               </spt:JobStateReasons>
               <spt:KOctetsProcessed>385</spt:KOctetsProcessed>
               <spt:MediaSheetsCompleted>4</spt:MediaSheetsCompleted>
               <spt:NumberOfDocuments>1</spt:NumberOfDocuments>
           </spt:JobStatus>
       </spt:JobStatusEvent>
   </wsoap12:Body>
</wsoap12:Envelope>
```

After the document has been printed, the laptop receives an event notification (using the SOAP-over-HTTP binding) from the printer.

```
<!-- Event notification about the status of the print job -->

POST /PrintEventSink
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.200:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<wsoap12:Envelope
       xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
       xmlns:spt="http://example.com/wsdp/sample/print">
   <wsoap12:Header>
       <wsa:To>http://192.168.0.200:80/PrintEventSink</wsa:To>
       <wsa:Action>
           http://example.com/wsdp/sample/print/JobEndStateEvent
       </wsa:Action>
       <wsa:MessageID>urn:uuid:2b1ef071-5bf6-6996-7ae8-654778a1917a</wsa:MessageID>
       <wse:Identifier>2597</wse:Identifier>
```

```
    </wsoap12:Header>
    <wsoap12:Body>
        <spt:JobEndStateEvent>
            <spt:JobEndState>
                <spt:JobId>1</spt:JobId>
                <spt:JobCompletedState>Completed</spt:JobCompletedState>
                <spt:JobCompletedStateReasons>
                    <spt:JobCompletedStateReason>
                        JobCompletedSuccessfully
                    </spt:JobCompletedStateReason>
                </spt:JobCompletedStateReasons>
                <spt:JobName>MyLetter</spt:JobName>
                <spt:JobOriginatingUserName>Mike.Fenelon</spt:JobOriginatingUserName>
                <spt:KOctetsProcessed>1235</spt:KOctetsProcessed>
                <spt:MediaSheetsCompleted>7</spt:MediaSheetsCompleted>
                <spt:NumberOfDocuments>2</spt:NumberOfDocuments>
            </spt:JobEndState>
        </spt:JobEndStateEvent>
    </wsoap12:Body>
</wsoap12:Envelope>
```

Later, when the user decides to shut down the laptop, the laptop unsubscribes itself from receiving event notifications from the printer. This is done by sending a WS-Eventing Unsubscribe message to the printer. The printer returns a WS-Eventing UnsubscribeResponse message. This ends the subscription. This message exchange is done using the SOAP-over-HTTP binding.

```
<!-- WS-Eventing Unsubscribe message -->

POST /SamplePrintService0
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.101:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
    <wsoap12:Header>
        <wsa: MessageID>urn:uuid:9t9dc791-2be6-4996-9af8-454778a1917a</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2004/08/eventing/Unsubscribe
        </wsa:Action>
        <wsa:To>http://192.168.0.101:80/SamplePrintService0</wsa:To>
        <wsa:ReplyTo>
            http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
        </wsa:ReplyTo>
        <wse:Identifier>
            urn:uuid:22e8a584-0d18-4228-b2a8-3716fa2097fa
        </wse:Identifier>
    </wsoap12:Header>
    <wsoap12:Body>
        <wse:Unsubscribe/>
    </wsoap12:Body>
```

```
</wsoap12:Envelope>
```

```
<!-- WS-Eventing UnsubscribeResponse message -->

HTTP/1.1 200
Content-type: application/soap+xml
Connection: close
Content-length: ...

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
    <wsoap12:Header>
        <wsa: MessageID>urn:uuid:4s8cd891-2be6-4996-9af8-454778a1917a</wsa:MessageID>
        <wsa: RelatesTo>urn:uuid:9t9dc791-2be6-4996-9af8-454778a1917a</wsa:RelatesTo>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2004/08/eventing/UnsubscribeResponse
        </wsa:Action>
        <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
    </wsoap12:Header>
    <wsoap12:Body/>
</wsoap12:Envelope>
```

The user turns off the printer. The printer sends an announcement indicating its departure from the network. This is done by sending a UDP multicast (using the SOAP-over-UDP binding) carrying a SOAP envelope that contains a WS-Discovery Bye message to the local subnet.

```
<!-- WS-Discovery multicast Bye message from the printer -->

<!--
SOAP-over-UDP message sent multicast from 192.168.0.101:65938 to 239.255.255.250:3702
-->

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:ac9087ee-6813-498c-9c1b-6272a22353f7</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/Bye
        </wsa:Action>
        <wsa:To>urn:schemas-xmlsoap-org:ws:2005:04:discovery</wsa:To>
        <wsdisco:AppSequence InstanceId="5094203799" MessageNumber="3"/>
    </wsoap12:Header>
    <wsoap12:Body>
        <wsdisco:Bye>
            <wsa:EndpointReference>
                <wsa:Address>urn:uuid:d10000d1-ca45-5fee-a376-112233445555</wsa:Address>
            </wsa:EndpointReference>
        </wsdisco:Bye>
    </wsoap12:Body>
</ wsoap12:Envelope>
```

3 Usage scenario: Service discovery using a proxy

This scenario (see Figure 3. Service discovery using a discovery proxyFigure 3**Error! Reference source not found.**) illustrates how to use a discovery proxy to discover services in a managed network. This scenario assumes the presence of a specialized network service, in this case, an enterprise repository acting as a discovery proxy.

The previous scenario (Usage scenario: Printing a document) used the ad hoc discovery mechanism, where the client multicasts a Probe message to discover matching services on the local network. In spite of its usefulness, the ad hoc discovery mechanism has some limitations:

- Network range of multicast messages is limited.
- Multicast messages increase network traffic.

To overcome those limitations, a specialized network service (discovery proxy) may be used. The discovery proxy typically stores the network location information of services that are present on the local subnet as well as on a wider network and allows for discovery of such services. Clients may directly probe the discovery proxy for matching services and thereby avoid generating multicast traffic.

Some ways for a client to obtain information about the discovery proxy:

- The client may send a multicast WS-Discovery Probe message explicitly looking for a discovery proxy on the network. In this case, the chances of finding a discovery proxy are limited by the multicast range allowed by the network.
- The client may get a unicast WS-Discovery Hello message from a discovery proxy in response to a multicast Probe message it had sent looking for a service. In this case, a discovery proxy that is within the multicast range allowed by the network may respond.
- The client may explicitly be pre-configured with the information about the discovery proxy.
- The client may discover the discovery proxy from a well known DHCP record containing the address of a discovery proxy.

The scenario consists of two services (Imaging Service and Print Service), an enterprise repository acting as a discovery proxy, and a Client (user with a laptop). This scenario assumes that the Client and the two services know about the discovery proxy via a well known DHCP record.

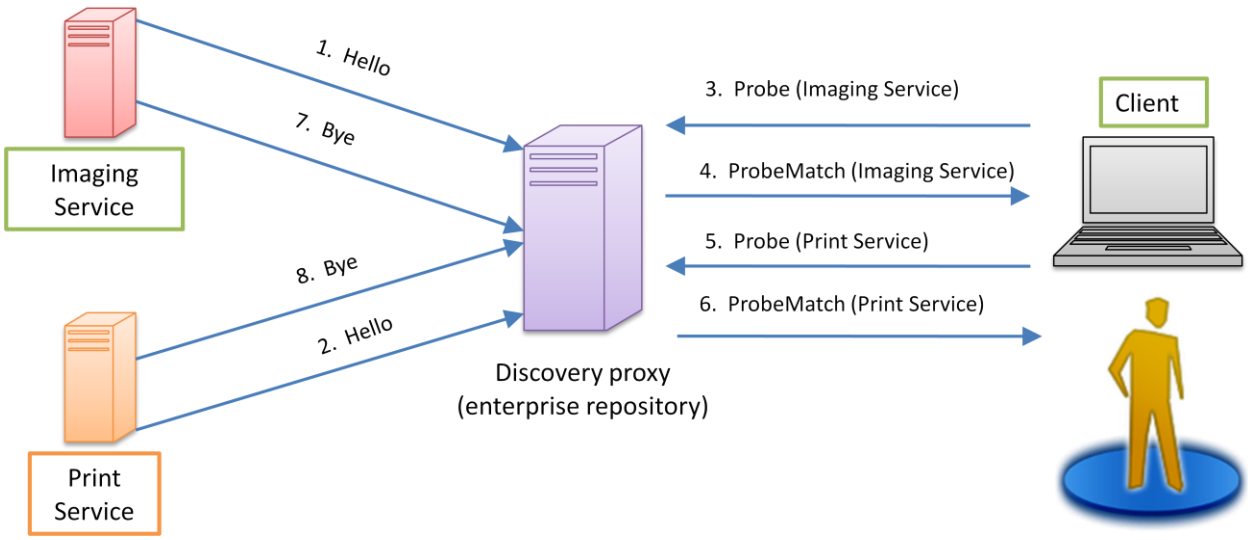

Figure 3. Service discovery using a discovery proxy

Here is a high-level description of the scenario:

1. When the devices or system hosting the Imaging Service joins the network, it announces its arrival to the discovery proxy via a unicast WS-Discovery Hello message. The discovery proxy receives the unicast Hello messages and stores the information about the Imaging Service.
2. Similarly, when the device or system hosting the Print Service joins the network, the discovery proxy receives information about the Print Service and stores it.
3. The Client (user with the laptop) enters the network. The user searches for the availability of an Imaging Service on the network. The laptop sends a unicast WS-Discovery Probe message to the discovery proxy searching for an Imaging Service on the network.
4. The discovery proxy responds with a unicast WS-Discovery Probe Match message carrying the network location information of the Imaging Service. The Client connects to the Imaging Service, obtains metadata information about the service, and eventually uses the service.
5. Later on, the user searches for the availability of a Print Service on the network. The laptop sends a unicast Probe message to the discovery proxy searching for a Print Service on the network.
6. The discovery proxy responds with a unicast Probe Match message carrying the network location information of the Print Service. The Client connects to the Print Service, obtains metadata information about the service, and eventually uses the service.
7. When the device or system hosting the Imaging Service leaves the network, it sends a unicast Bye message to the discovery proxy. The discovery proxy removes information about the Imaging Service from its internal store.
8. Similarly, when the device or system hosting the Print Service leaves the network, the discovery proxy receives information about the departure and removes it from its internal store.

The following is a more detailed description of this scenario. It contains sample message exchanges between the Client, discovery proxy, Imaging Service, and Print Service.

3.1 Detailed description

When the device hosting the Imaging Service enters the network, it announces its arrival via a unicast WS-Discovery Hello message sent directly to the discovery proxy. Similarly, when the device hosting the Print Service enters the network, it sends a unicast Hello message directly to the discovery proxy. This is done using the SOAP-over-HTTP binding. The discovery proxy receives the unicast Hello messages and stores the information about the Imaging Service and Print Service.

```
<!-- WS-Discovery unicast Hello message from Imaging Service to discovery proxy -->

POST /sample/discoproxy/proxy
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.2:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery"
    xmlns:imx="http://example.com/sample/imaging">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:ac9823ee-6813-498c-8c1b-6272a22353f7</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/Hello
        </wsa:Action>
        <wsa:To>http://192.168.0.2:80/sample/discoproxy/proxy</wsa:To>
        <wsdisco:AppSequence InstanceId="7094203799" MessageNumber="1"/>
    </wsoap12:Header>
    <wsoap12:Body>
        <wsdisco:Hello>
            <wsa:EndpointReference>
                <wsa:Address>urn:uuid:e10000d1-ca45-7fee-a376-112233445555</wsa:Address>
            </wsa:EndpointReference>
            <wsdisco:Types>imx:ImagingDeviceType</wsdisco:Types>
            <wsdisco:XAddrs>
                http://192.168.0.150:80/sample/imaging/ImagingService
            </wsdisco:XAddrs>
            <wsdisco:MetadataVersion>1</wsdisco:MetadataVersion>
        </wsdisco:Hello>
    </wsoap12:Body>
</wsoap12:Envelope>
```

```
<!-- WS-Discovery unicast Hello message from Print Service to discovery proxy -->

POST /sample/discoproxy/proxy
HTTP/1.1
```

```
Content-Type: application/soap+xml
Host: 192.168.0.2:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery"
    xmlns:prx="http://example.com/sample/print">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:bd9823ee-6813-498c-8c1b-6272a22353f7</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/Hello
        </wsa:Action>
        <wsa:To>http://192.168.0.2:80/sample/discoproxy/proxy</wsa:To>
        <wsdisco:AppSequence InstanceId="9094203799" MessageNumber="1"/>
    </wsoap12:Header>
    <wsoap12:Body>
        <wsdisco:Hello>
            <wsa:EndpointReference>
                <wsa:Address>urn:uuid:f90000d1-ca45-7fee-a376-112233445555</wsa:Address>
            </wsa:EndpointReference>
            <wsdisco:Types>prx:PrintDeviceType</wsdisco:Types>
            <wsdisco:XAddrs>http://192.168.0.160:80/sample/print/PrintService</wsdisco:XAddrs>
            <wsdisco:MetadataVersion>1</wsdisco:MetadataVersion>
        </wsdisco:Hello>
    </wsoap12:Body>
</wsoap12:Envelope>
```

The Client (user with the laptop) enters the network. The user searches for the availability of an Imaging Service on the network. The laptop sends a unicast WS-Discovery Probe message to the discovery proxy searching for an Imaging Service on the network. The discovery proxy responds with a unicast WS-Discovery Probe Match message carrying the network location information of the Imaging Service. Both the Probe and Probe Match messages are sent using SOAP-over-HTTP binding. The Client connects to the Imaging Service, obtains metadata information about the service, and eventually uses the service. This is similar to the steps described in section 2, Usage scenario: Printing a document.

```
<!-- WS-Discovery unicast Probe message from Client to discovery proxy-->

POST /sample/discoproxy/proxy
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.2:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
```

```
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery"
    xmlns:imx="http://example.com/sample/imaging">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:ac9855ff-6813-498c-8c1b-6272a22353f7</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/Probe
        </wsa:Action>
        <wsa:To>http://192.168.0.2:80/sample/discoproxy/proxy</wsa:To>
        <wsa:ReplyTo>http://192.168.0.140:80/client</wsa:ReplyTo>
    </wsoap12:Header>
    <wsoap12:Body>
        <wsdisco:Probe>
            <wsdisco:Types>imx:ImagingDeviceType</wsdisco:Types>
        </wsdisco:Probe>
    </wsoap12:Body>
</wsoap12:Envelope>
```

```
<!-- WS-Discovery unicast Probe Match message from discovery proxy -->

POST /sample/imaging
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.140:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery"
    xmlns:imx="http://example.com/sample/imaging">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:1b7dc791-2be6-4991-9af1-454778a1917a</wsa:MessageID>
        <wsa:RelatesTo>urn:uuid:ac9855ff-6813-498c-8c1b-6272a22353f7</wsa:RelatedTo>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/ProbeMatches
        </wsa:Action>
        <wsa:To>http://192.168.0.140:80/client</wsa:To>
        <wsdisco:AppSequence InstanceId="5094203799" MessageNumber="1"/>
    </wsoap12:Header>
    <soap12:Body>
        <wsdisco:ProbeMatches>
            <wsdisco:ProbeMatch>
                <wsa:EndpointReference>
                    <wsa:Address>urn:uuid:e10000D1-ca45-7fee-a376-112233445555</wsa:Address>
                </wsa:EndpointReference>
                <wsdisco:Types>imx:ImagingDeviceType</wsdisco:Types>
                <wsdisco:XAddrs>
                    http://192.168.0.150:80/sample/imaging/ImagingService
                </wsdisco:XAddrs>
                <wsdisco:MetadataVersion>1</wsdisco:MetadataVersion>
            </wsdisco:ProbeMatch>
        </wsdisco:ProbeMatches>
    </wsoap12:Body>
</wsoap12:Envelope>
```

Later on, the user searches for the availability of a Print Service on the network. The laptop sends a unicast WS-Discovery Probe message to the discovery proxy searching for a Print Service on the network. The discovery proxy responds with a unicast WS-Discovery Probe Match message carrying the network location information of the Print Service. Both the Probe and Probe Match messages are sent using SOAP-over-HTTP binding. The Client connects to the Print Service, obtains metadata information about the service, and eventually uses the service. This is similar to the steps described in section 2, Usage scenario: Printing a document.

```
<!-- WS-Discovery unicast Probe message from Client to discovery proxy -->

POST /sample/discoproxy/proxy
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.2:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery"
    xmlns:prx="http://example.com/sample/print">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:bd8966ff-6813-498c-8c1b-6272a22353f7</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/Probe
        </wsa:Action>
        <wsa:To>http://192.168.0.2:80/sample/discoproxy/proxy</wsa:To>
        <wsa:ReplyTo>http://192.168.0.140:80/client</wsa:ReplyTo>
    </wsoap12:Header>
    <wsoap12:Body>
        <wsdisco:Probe>
            <wsdisco:Types>prx:PrintDeviceType</wsdisco:Types>
        </wsdisco:Probe>
    </wsoap12:Body>
</wsoap12:Envelope>
```

```
<!-- WS-Discovery unicast Probe Match message from discovery proxy -->

POST /sample/print
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.140:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
```

```
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery"
    xmlns:prx="http://example.com/sample/print">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:8e4cy791-2be6-4991-9af1-454778a1917a</wsa:MessageID>
        <wsa:RelatesTo>urn:uuid:bd8966ff-6813-498c-8c1b-6272a22353f7</wsa:RelatedTo>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/ProbeMatches
        </wsa:Action>
        <wsa:To>http://192.168.0.140:80/client</wsa:To>
        <wsdisco:AppSequence InstanceId="5094203799" MessageNumber="2"/>
    </wsoap12:Header>
    <soap12:Body>
        <wsdisco:ProbeMatches>
            <wsdisco:ProbeMatch>
                <wsa:EndpointReference>
                    <wsa:Address>urn:uuid:f90000d1-ca45-7fee-a376-112233445555</wsa:Address>
                </wsa:EndpointReference>
                <wsdisco:Types>prx:PrintDeviceType</wsdisco:Types>
                <wsdisco:XAddrs>
                    http://192.168.0.160:80/sample/print/PrintService
                </wsdisco:XAddrs>
                <wsdisco:MetadataVersion>1</wsdisco:MetadataVersion>
            </wsdisco:ProbeMatch>
        </wsdisco:ProbeMatches>
    </wsoap12:Body>
</wsoap12:Envelope>
```

When the device or system hosting the Imaging Service leaves the network, it sends a unicast Bye message to the discovery proxy. The discovery proxy removes information about the Imaging Service from its internal store. Similarly, when the device or system hosting the Print Service leaves the network, it sends a unicast Bye message to the discovery proxy. The discovery proxy removes information about the Print Service from its internal store. The Bye messages are sent using SOAP-over-HTTP binding.

```
<!-- WS-Discovery unicast Bye message from Imaging Service -->

POST /sample/discoproxy/proxy
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.2:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:ef8923ee-6813-498c-8c1b-6272a22353f7</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/Bye
        </wsa:Action>
        <wsa:To>http://192.168.0.2:80/sample/discoproxy/proxy</wsa:To>
        <wsdisco:AppSequence InstanceId="7094203799" MessageNumber="2"/>
    </wsoap12:Header>
```

```
    <wsoap12:Body>
        <wsdisco:Bye>
            <wsa:EndpointReference>
                <wsa:Address>urn:uuid:e10000d1-ca45-7fee-a376-112233445555</wsa:Address>
            </wsa:EndpointReference>
        </wsdisco:Bye>
    </wsoap12:Body>
</wsoap12:Envelope>
```

```
<!-- WS-Discovery unicast Bye message from Print Service -->

POST /sample/discoproxy/proxy
HTTP/1.1
Content-Type: application/soap+xml
Host: 192.168.0.2:80
Content-Length: ...
Connection: Keep-Alive
Cache-Control: no-cache
Pragma: no-cache

<?xml version="1.0" encoding="utf-8"?>
<wsoap12:Envelope
    xmlns:wsoap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdisco="http://schemas.xmlsoap.org/ws/2005/04/discovery">
    <wsoap12:Header>
        <wsa:MessageID>urn:uuid:zp5723ee-6813-498c-8c1b-6272a22353f7</wsa:MessageID>
        <wsa:Action>
            http://schemas.xmlsoap.org/ws/2005/04/discovery/Bye
        </wsa:Action>
        <wsa:To>http://192.168.0.2:80/sample/discoproxy/proxy</wsa:To>
        <wsdisco:AppSequence InstanceId="9094203799" MessageNumber="2"/>
    </wsoap12:Header>
    <wsoap12:Body>
        <wsdisco:Bye>
            <wsa:EndpointReference>
                <wsa:Address>urn:uuid:f90000d1-ca45-7fee-a376-112233445555</wsa:Address>
            </wsa:EndpointReference>
        </wsdisco:Bye>
    </wsoap12:Body>
</wsoap12:Envelope>
```

4 Implementation guidance on using WS-Discovery with multi-homed devices and systems

This section provides implementation guidance on the use of WS-Discovery with multi-homed devices and systems.

4.1 Multi-homed devices and systems

It is possible that a Client, Target Service or a Discovery Proxy (these terms are defined in **[WS-DISCOVERY]**) may be multi-homed – that is, they may support multiple network interfaces. A multi-homed Target Service is accessible via multiple transport

addresses and a multi-homed client may access Target Services on different networks. Since there is no single transport address that is valid across multiple networks, transport addresses used on one network cannot be used on another network. The following is implementation guidance on the use of WS-Discovery with Clients, Target Services, and Discovery Proxies that support multiple network interfaces.

Figure 4 depicts a typical scenario where some Clients and Target Services have multiple network interfaces.

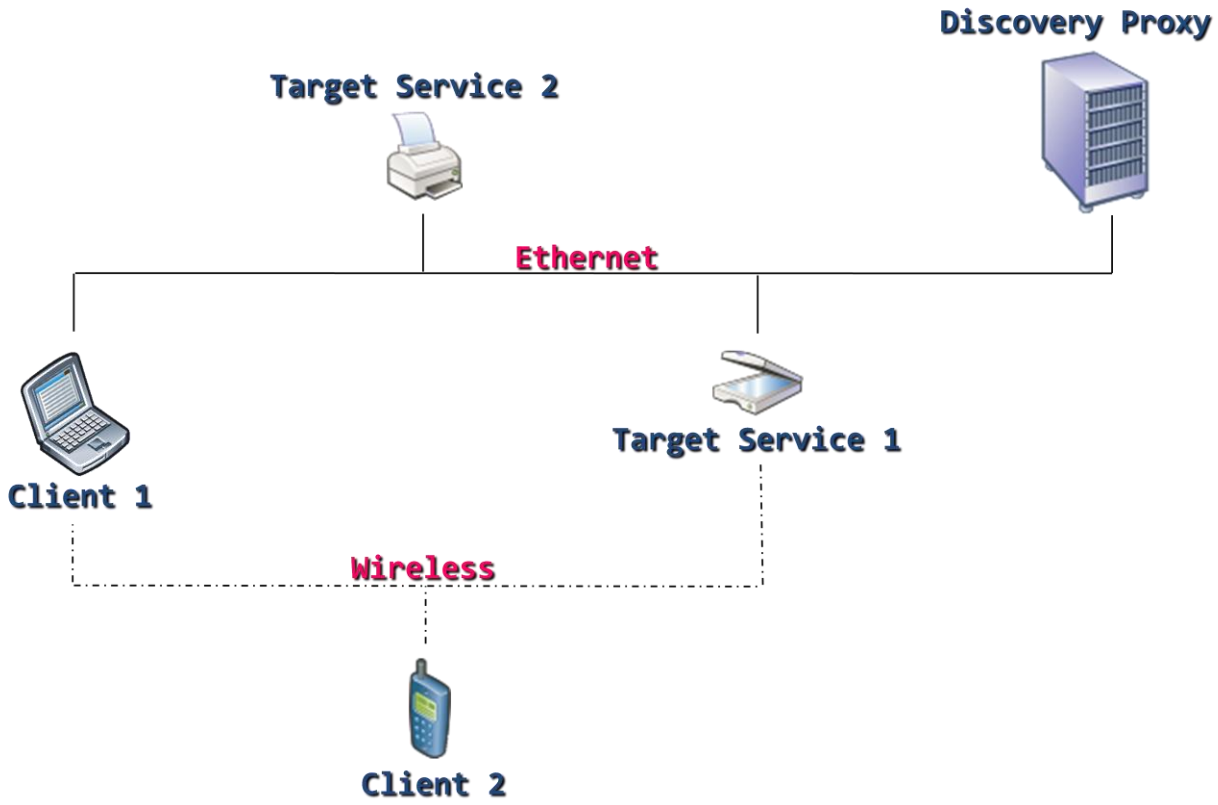

Figure 4. Multiple network interface scenario

Client 1 has a network interface for both an Ethernet and a wireless network. To discover all Target Services, Client 1 needs to send a Probe (or Resolve) over all of its active network interfaces. For example, if Client 1 sends a Probe over only the wireless network, it will not be able to discover Target Service 2.

If a Client detects a Discovery Proxy on a network, it may use that Discovery Proxy on that network while multicasting on others. For example, Client 1 will use the Discovery Proxy for the Ethernet network but will multicast for the wireless network.

Target Service 1 is also on both the Ethernet and wireless network. When it receives a Probe (or Resolve), Target Service 1 must be careful to return reachable transport addresses in a Probe Match (or Resolve Match) or else some Clients could discover the Target Service but be unable to communicate with it. For instance, if Target Service 1 responded to a Probe (or Resolve) from Client 2 with a transport address for the Ethernet LAN, Client 2 would not be able to send a unicast message to it.

4.2 Implementation guidance

4.2.1 General

1. A Target Service (or Discovery Proxy) should not assume that a Client is available on all the multiple networks it is on or that the Client is available on only a specific network.

2. A Client should not assume that a Target Service (or Discovery Proxy) is available on all the multiple networks it is on or that the Target Service or Discovery Proxy is available on only a specific network.

3. A Target Service (or Discovery Proxy) should use the same WS-Addressing [address] and [reference parameters] properties for all messages over all network interfaces. This allows a Client to recognize a Target Service (or Discovery Proxy) when they share multiple networks.

4. A Target Service (or Discovery Proxy) must maintain a single WS-Discovery MetadataVersion across all network interfaces. This allows a Client that shares multiple networks with the Target Service (or Discovery Proxy) to maintain a consistent view of the Target Service (or Discovery Proxy) metadata.

5. An implementation may choose to behave as a Client, Target Service, or Discovery Proxy over only some of its network interfaces. This allows an implementation to use WS-Discovery over whichever network interfaces that are appropriate.

6. If an implementation exposes different capabilities over two network interfaces, it must expose itself as a distinct Target Service on each network interface; that is, as one Target Service on one network and a second Target Service on the other network. This allows a Client to correctly associate capabilities, transport addresses, etc. with a Target Service. Any higher-level identification between Target Services exposed by a common underlying implementation is an implementation detail.

7. In messages containing transport addresses sent by a Target Service (or Discovery Proxy) to a Client, in order to ensure that the Target Service (or Discovery Proxy) is reachable by the Client, the Target Service (or Discovery Proxy) should use the transport address corresponding to the network interface used to send those messages.

4.2.2 Hello and Bye

1. When a Target Service (or Discovery Proxy) sends a Hello (or Bye) over multiple network interfaces, the messages for all interfaces should be

transmitted within APP_MAX_DELAY (as defined in **[WS-DISCOVERY]**) to help Clients listening on more than one of these interfaces to select the most appropriate network interface.

4.2.3 Probe and Resolve

1. A Client must send a Probe (or Resolve) with a **[reply endpoint]** property of either "http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous" or a transport address reachable by the network interface used to send the Probe (or Resolve). This ensures that the **[reply endpoint]** has a transport address that is reachable by a matching Target Service (or Discovery Proxy) that receives the Probe (or Resolve).

2. A multi-homed Target Service (or Discovery Proxy) should send Probe Match and Resolve Match messages to the WS-Addressing [reply endpoint] property specified in the corresponding Probe or Resolve message.

5 Conclusion

There are a number of possible usage scenarios for the DPWS, WS-Discovery and SOAP-over-UDP specifications. This document illustrated some useful scenarios to describe and motivate the use of those specifications. For a detailed description of the protocols and semantics, refer to the DPWS, WS-Discovery and SOAP-over-UDP specifications.

The DPWS, WS-Discovery, and SOAP-over-UDP specifications use, compose with, and build upon other Web Services specifications. WS-Discovery provides a dynamic discovery protocol that works in different types of networks. DPWS brings the rich functionality of Web Services to resource constrained devices. This enables use of existing tools provided by Web Service platform providers for developing DPWS based devices and easier integration with existing Web Service deployments.

Appendix A. Acknowledgements

This document has been developed as a result of joint work with many individuals and teams from Microsoft, including Dan Conti, Colleen Evans, Asir Vedamuthu, Greg Carpenter, Doug Bunting, Krishnan Gopalan, Michael champion, Anurag Pandit, Ashay Chaudhary, Piyush Joshi, and Jon Cole; this work was motivated and made possible by Paul Cotton.

Appendix B. XML namespaces

Table 1 lists XML namespaces that are used in this document. The choice of any namespace prefix is arbitrary and not semantically significant. The prefixes below have several characters in them to make them more readable. However, in practice, it

is better to use a single character for prefixes and thereby avoid unnecessarily increasing the message size; this is particularly important for multicast messages.

Table 1. Prefixes and XML namespaces used in this document

Prefix	XML namespace	Specification(s)
wsdp	http://schemas.xmlsoap.org/ws/2006/02/devprof	**[DPWS]**
wsdisco	http://schemas.xmlsoap.org/ws/2005/04/discovery	**[WS-DISCOVERY]**
wsoap12	http://www.w3.org/2003/05/soap-envelope	**[SOAP]**
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing	**[WS-Addressing]**
wsdl	http://schemas.xmlsoap.org/wsdl/	**[WSDL]**
wse	http://schemas.xmlsoap.org/ws/2004/08/eventing	**[WS-Eventing]**
wsp	http://schemas.xmlsoap.org/ws/2004/09/policy	**[WS-Policy]**
mex	http://schemas.xmlsoap.org/ws/2004/09/mex	**[WS-MEX]**
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	**[WS-Security]**

Appendix C. References

[DPWS] Devices Profile for Web Services (DPWS)
http://schemas.xmlsoap.org/ws/2006/02/devprof

[WS-DISCOVERY] Web Services Discovery (WS-Discovery)
http://schemas.xmlsoap.org/ws/2005/04/discovery

[SOAP-over-UDP] SOAP-over-UDP
http://schemas.xmlsoap.org/ws/2004/09/soap-over-udp

[SOAP-over-HTTP] SOAP-over-HTTP binding – SOAP 1.2 Part 2, Section 7
http://www.w3.org/TR/2003/REC-soap12-part2-20030624/#soapinhttp

[UDP] User Datagram Protocol (UDP)
http://www.ietf.org/rfc/rfc768.txt

[HTTP] Hypertext Transfer Protocol (HTTP) 1.1
http://www.ietf.org/rfc/rfc2616.txt

[SOAP] Simple Object Access Protocol (SOAP) 1.2
http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

[WS-Addressing] Web Services Addressing (WS-Addressing)
http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/

[MTOM] SOAP Message Transmission Optimization Mechanism (MTOM)
http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/

[WSDL] Web Services Description Language (WSDL) 1.1

http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[WS-Policy] Web Services Policy (WS-Policy)
http://schemas.xmlsoap.org/ws/2004/09/policy

[WS-MEX] Web Services Metadata Exchange (WS-MetadataExchange)
http://schemas.xmlsoap.org/ws/2004/09/mex/

[WS-Transfer] Web Services Transfer (WS-Transfer)
http://schemas.xmlsoap.org/ws/2004/09/transfer/

[WS-Eventing] Web Services Eventing (WS-Eventing)
http://schemas.xmlsoap.org/ws/2004/08/eventing/

[WS-Security] Web Services Security
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf