[MS-BPDP]:
Background Intelligent Transfer Service (BITS) Peer-Caching:
Peer Discovery Protocol Specification

Intellectual Property Rights Notice for Protocol Documentation

- This protocol documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the protocols, and may distribute portions of it in your implementations of the protocols or your documentation as necessary to properly document the implementation. This permission also applies to any documents that are referenced in the protocol documentation.
- Microsoft does not claim any trade secret rights in this documentation.
- Microsoft has patents that may cover your implementations of the protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. If you are interested in obtaining a patent license, please contact protocol@microsoft.com.
- The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

This protocol documentation is intended for use in conjunction with publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

A protocol specification does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them.

Revision Summary

Date	Revision History	Revision Class	Comments
04/10/2007	1.0		Version 1.0 release
05/18/2007	1.2		Version 1.2 release
06/08/2007	1.2.1	Editorial	Revised and edited the technical content.

Copyright © 2008 Microsoft Corporation.

Date	Revision History	Revision Class	Comments
07/10/2007	1.3	Minor	Updated the technical content.
08/17/2007	1.3.1	Editorial	Revised and edited the technical content.
09/21/2007	1.3.2	Editorial	Revised and edited the technical content.
10/26/2007	1.3.3	Editorial	Revised and edited the technical content.
01/25/2008	1.3.4	Editorial	Revised and edited the technical content.

Copyright © 2008 Microsoft Corporation.

Table of Contents

Copyright © 2008 Microsoft Corporation.

Copyright © 2008 Microsoft Corporation.

1 Introduction

This document defines a Microsoft proprietary protocol referred to as the Background Intelligent Transfer Service (BITS) Peer-Caching: Peer Discovery Protocol . This protocol is used to locate hosts in a **domain** that supports the URL-caching protocol implemented by BITS. The protocol is implemented by using the Web Services Dynamic Discovery (WS-Discovery) Protocol, as specified in [WS-Discovery].

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Domain
Fully Qualified Domain Name (FQDN)
Globally Unique Identifier (GUID)

The following terms are specific to this document:

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC3513] Hinden, R. and Deering, S., "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003, http://www.ietf.org/rfc/rfc3513.txt

[SOAP1.2/1] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624

[SOAP-UDP] Combs, H., et al., "SOAP-over-UDP", September 2004, http://specs.xmlsoap.org/ws/2004/09/soap-over-udp/soap-over-udp.pdf

If you have any trouble finding [SOAP-UDP], please check here.

[WS-Discovery] Beatty, J. et. al, "Web Services Dynamic Discovery (WS-Discovery)", April 2005, http://www1.webmethods.com/PDF/WS-Discovery.pdf

If you have any trouble finding [WS-Discovery], please check here.

[WSAddressing] Box, D. et al., "Web Services Addressing (WS-Addressing)", August 2004, http://www.w3.org/Submission/ws-addressing/

Copyright © 2008 Microsoft Corporation.

If you have any trouble finding [WSAddressing], please check here.

[WSD-Schema] Microsoft Corporation, "WS-Discovery Specification Schema", 2004, http://schemas.xmlsoap.org/ws/2004/10/discovery/

If you have any trouble finding [WSD-Schema], please check here.

1.2.2 Informative References

[MSDN-BITS] Microsoft Corporation, "Background Intelligent Transfer Service", http://msdn2.microsoft.com/en-us/library/aa362827.aspx

[MS-BPAU] Microsoft Corporation, "Background Intelligent Transfer Service (BITS) Peer-Caching: Peer Authentication Protocol Specification", April 2007.

[MS-BPCR] Microsoft Corporation, "Background Intelligent Transfer Service (BITS) Peer-Caching: Content Retrieval Protocol Specification", April 2007.

1.3 Protocol Overview (Synopsis)

The BITS Peer-Caching: Peer Discovery Protocol is used to locate networked hosts or devices that are implementing the server role of the BITS Peer-Caching: Content Retrieval Protocol. The BITS Peer-Caching: Peer Discovery Protocol provides a way for peer servers to announce their presence to connected subnets and a way for peer clients to locate servers in connected subnets.

The BITS Peer-Caching: Peer Discovery Protocol is a specialization of Web Services Dynamic Discovery (WS-Discovery), as specified in [WS-Discovery], and follows its model for announcing and locating resources. The protocol defines a client role and a server role. A server announces its presence to connected IP subnets via a multicasted Hello message to UDP port 3702. A client discovers servers passively by listening for Hello messages. A client may also solicit for servers by multicasting a Probe message to the same UDP port; servers with matching characteristics reply to the client with unicast Probe-Match messages.

Windows uses the BITS Peer-Caching: Peer Discovery Protocol to implement a distributed peer-to-peer cache of URL content for use by the Background Intelligent Transfer Service (BITS) component. For more information about BITS, see [MSDN-BITS].

1.4 Relationship to Other Protocols

The BITS Peer-Caching: Peer Discovery Protocol does not authenticate computers to each other; Windows uses the Background Intelligent Transfer Service (BITS) Peer-Caching: Peer Authentication Protocol Specification for mutual authentication of potential peers. For more information, see [MS-BPAU].

WS-Discovery, and therefore the BITS Peer-Caching: Peer Discovery Protocol, uses SOAP-over-UDP as its network transport. For more information, see [SOAP-UDP].

A host implementing the client or server role of the BITS Peer-Caching: Peer Discovery Protocol typically also implements the same role of the BITS Peer-Caching: Content Retrieval Protocol.

1.5 Prerequisites/Preconditions

This protocol defines no prerequisites.

Copyright © 2008 Microsoft Corporation.

1.6 Applicability Statement

The primary purpose of the BITS Peer-Caching: Peer Discovery Protocol is to locate peer servers for use by the BITS Peer-Caching: Content Retrieval Protocol, as specified in [MS-BPCR]. The BITS Peer-Caching: Peer Discovery Protocol is intended for use by hosts that are members of a Windows domain.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Protocol Versions**: A server advertises the versions of the protocol it supports via Hello and Probe-Match messages.

  A client does not advertise the protocol versions it supports.

  This specification defines version 1 of the BITS Peer-Caching: Peer Discovery Protocol. The format of each message in version 1 is defined in section 2.2 and its subsidiary sections. Because this is the initial release of the protocol, no additional versions are defined at time of publication.

- **Capability Negotiation**: The BITS Peer-Caching: Peer Discovery Protocol implicitly allows negotiation of additional capabilities by the presence or absence of additional XML elements in each message. Version 1 does not define any capabilities.

1.8 Vendor-Extensible Fields

No vendor-extensible fields are defined.

1.9 Standards Assignments

The following XML namespaces are used in this section.

Prefix	XML namespace	Specification
msbits	http://schemas.microsoft.com/windows/2005/05/BITS/cache	This specification ([MS-BPDP])
a	http://schemas.xmlsoap.org/ws/2004/08/addressing	[WSAddressing]
d	http://schemas.xmlsoap.org/ws/2005/04/discovery	[WS-Discovery]

Copyright © 2008 Microsoft Corporation.

2 Messages

The following sections specify the message syntax for the BITS Peer-Caching: Peer Discovery Protocol.

2.1 Transport

The BITS Peer-Caching: Peer Discovery Protocol is a specialization of the WS-Discovery Protocol. The Web Services Dynamic Discovery (WS-Discovery) Protocol, including dependent transports, is as specified in [WS-Discovery].

2.2 Message Syntax

The BITS Peer-Caching: Peer Discovery Protocol follows the message syntax of WS-Discovery. Messages MUST use SOAP version 1.2, as specified in [SOAP1.2/2]. This protocol does not require the generation or processing of SOAP faults. No additional SOAP faults are defined.

The BITS Peer-Caching: Peer Discovery Protocol defines the http://schemas.microsoft.com/windows/2005/05/BITS/cache namespace as follows:

- The element **msbits:Fqdn** MUST contain the fully qualified DNS domain name of a host.
- The element **msbits:version** MUST contain a list of versions of the BITS Peer-Caching: Peer Discovery Protocol that are supported by the sender. Versions MUST be unsigned 32-bit integers; the list is delimited by white space. If version 1 is an element of the list, it MUST be the first element. This specification defines version 1; no other versions are defined at time of publication.
- The type **msbits:PeerServer** represents an instance of the server role.

[WS-Discovery] section 2.6 refers to the **a:EndpointReference** element, as specified in [WSAddressing] section 2. Within messages sent by the BITS Peer-Caching: Peer Discovery Protocol, the specification of the **a:EndpointReference** element is constrained in the following ways.

- The **a:Address** child element follows the recommendation of "uuid:" followed by a **GUID**. This MUST be the instance GUID of the server referred to in section 3.1.1.1.

A transport address in the **d:XAddrs** element MUST be in one of the following formats.

```
IPV4-str= "https://" ipv4-address
IPV6-str= "https://[" ipv6-address "]"
```

Where:

- The Ipv4-address MUST be a dotted IPv4 address.
- The Ipv6-address MUST be one of the formats, as specified in [RFC3513] section 2.2.

The following sections specify requirements for each of the messages sent by the BITS Peer-Caching: Peer Discovery Protocol.

Copyright © 2008 Microsoft Corporation.

2.2.1 Hello Message

A server of the BITS Peer-Caching: Peer Discovery Protocol uses the Hello message to announce its presence.

The format of the Hello message is as specified in [WS-Discovery] section 4.1. The following additional constraints are placed on the **/s:Envelope/s:Body/d:Hello** element:

- The **a:EndpointReference** child element MUST conform to section 2.2 of this document. It MUST contain exactly one msbits:Fqdn child element and exactly one msbits:version child element. These elements carry metadata about the server even though they are child elements of the a:EndpointReference child element, and SHOULD NOT be copied into subsequent messages addressed to the Endpoint Reference in question.
- A single **d:Types** child element MUST be present and include the type "msbits:PeerServer".
- A single **d:Scopes** child element MUST be present and contain at least one scope.<1>
- A single **d:XAddrs** child element MUST be present. The URI list MUST contain at least one IP address in each of the subnets to which the message is sent.<2> For the format of address URIs, see section 2.2.

2.2.2 Bye Message

A server of the BITS Peer-Caching: Peer Discovery Protocol uses the Bye message to indicate that it is disconnecting from a network.

The format of the Bye message is as specified in [WS-Discovery] section 4.1.

The BITS Peer-Caching: Peer Discovery Protocol places no new requirements on the Bye message.

2.2.3 Probe Message

A client of the BITS Peer-Caching: Peer Discovery Protocol uses the Probe message to solicit potential servers.

The format of the Probe message is as specified in [WS-Discovery] section 5.2. The following additional constraints are placed on the **/s:Envelope/s:Body/d:Probe** element:

- A single **d:Types** child element MUST be present and include the type "msbits:PeerServer".
- A single **d:Scopes** child element MUST be present and contain at least one scope.<3>

A client or server MUST support the "http://schemas.xmlsoap.org/ws/2005/04/discovery/rfc2396" matching rule and MAY support other rules.

2.2.4 Probe-Match Message

A server of the BITS Peer-Caching: Peer Discovery Protocol sends a Probe-Match message after it receives a Probe message that matches the server's WS-Discovery type and scope.

The format of the Probe-Match message is as specified in [WS-Discovery] section 5.3. The following additional constraints are placed on the **/s:Envelope/s:Body/d:Probe-Match** element:

- The **a:EndpointReference** child element MUST conform to section 2.2 of this document. It MUST contain exactly one msbits:Fqdn child element and exactly one msbits:version child element. These elements carry metadata about the server even though they are child elements

Copyright © 2008 Microsoft Corporation.

of the a:EndpointReference child element, and SHOULD NOT be copied into subsequent messages addressed to the Endpoint Reference in question.

- A single **d:Types** child element MUST be present and include the type "msbits:PeerServer".
- A single **d:Scopes** child element MUST be present and contain at least one scope conforming to section 2.2 of this document.<4>
- A single **d:XAddrs** child element MUST be present and its URI list MUST contain at least one address in the subnet to which the message is sent.<5>

2.2.5 Other WS-Discovery Messages

The BITS Peer-Caching: Peer Discovery Protocol does not use WS-Discovery's Resolve and Resolve-Match messages and makes no changes to the definitions as specified in [WS-Discovery] sections 6.1 and 6.2. They SHOULD NOT be sent by an implementation of this protocol, and SHOULD be ignored or treated as not matching if received.<6>

Copyright © 2008 Microsoft Corporation.

3 Protocol Details

3.1 Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The server behaves as a WS-Discovery Target Service, as specified in [WS-Discovery].

3.1.1.1 Protocol Metadata

The server maintains several pieces of metadata about itself:

- Its **FQDN**.
- Its active WS-Discovery scopes.
- An unsigned 32-bit integer representing the WS-Discovery metadata version.
- A GUID that is the instance GUID for WS-Discovery.

3.1.1.2 Table of Connected Subnets

The server maintains a table of all subnets for which it currently has an active IP address. Each row of the table represents a single subnet and includes the following data:

- A list of server IP addresses in the subnet.
- The subnet ID.
- The subnet mask (for IPv4) or prefix (for IPv6).

Note The abstract data model can be implemented in a variety of ways. This protocol does not prescribe or advocate any specific implementation technique.

3.1.2 Timers

No protocol-specific timers are required by the server.

3.1.3 Initialization

The first time the protocol is initialized, the server MUST create a unique instance GUID and SHOULD set its metadata version to zero. In subsequent initializations, the server SHOULD test whether its set of IP addresses has changed the previous invocation by using the metadata version from the previous invocation if the set is unchanged or incrementing the metadata version if the set is different. Otherwise, the server MUST create a new unique instance GUID.<7>

The server MUST identify its WS-Discovery scopes by implementation-dependent means.<8>

The server MUST begin listening for messages. Transport information is specified in [WS-Discovery] section 2.4.

Copyright © 2008 Microsoft Corporation.

3.1.4 Higher-Layer Triggered Events

3.1.4.1 Shutdown

The server MUST ignore further incoming messages.

The server MAY send a Bye message as specified in [WS-Discovery] sections 3 and 4.2.<9>

The server SHOULD close the network ports specified in [WS-Discovery] section 2.4.<10>

3.1.5 Message Processing Events and Sequencing Rules

The server MUST verify that each received message matches the schema, as specified in [WS-Discovery] Appendix III, discarding malformed messages. Further parsing depends on the message type.

3.1.5.1 Probe

The server adheres to the requirements, as specified in [WS-Discovery] section 5. In addition, BITS Peer-Caching: Peer Discovery Protocol adds the following requirements.

When the server receives a Probe message, it MUST verify that it satisfies the requirements in sections 2.2 and 2.2.3; if not, the message MUST be discarded.

When a matching Probe message is received, the server MUST reply with a Probe-Match message, following the rules as specified in [WS-Discovery] section 5.3 and in section 2.2.4 of this document.

When a Probe-Match message is sent to a particular subnet, the **/s:Envelope/s:Body/d:Probe-Match/d:XAddrs** element MUST contain the server addresses in that subnet from the table of connected subnets. The message SHOULD NOT contain addresses in other subnets.<11> For the URI encoding of server addresses, see section 2.2.

The server MAY limit the number and rate of Probe messages processed.

3.1.5.2 Resolve

A server is required to respond to a Resolve message that matches its Endpoint Reference, as specified in [WS-Discovery] section 6.1. The BITS Peer-Caching: Peer Discovery Protocol relaxes that requirement.

When the server receives a Resolve message, it MAY ignore the message.<12> If the server chooses to process Resolve messages, it MUST follow the rules, as specified in [WS-Discovery] section 6.2.

In WS-Discovery, the Resolve/Resolve-Match message exchange is used when a client knows the Endpoint Reference of a server but not its XAddrs, and it needs the XAddrs. All Hello and Probe-Match messages sent by the BITS Peer-Caching: Peer Discovery Protocol server role carry the server's XAddrs in addition to the Endpoint Reference, so the additional message exchange is not necessary.

3.1.5.3 Other Message Types

All other messages are discarded without further processing.

Copyright © 2008 Microsoft Corporation.

3.1.6 Timer Events

Timers are defined in [WS-Discovery] sections 2.4, 3, and 7.

BITS Peer-Caching: Peer Discovery Protocol defines no additional requirements.

3.1.7 Other Local Events

3.1.7.1 Add a Local IP Address

When a local IP address is added, the server checks whether it is a loopback address or a temporary IPv6 address. If it is either address, the notification MUST be ignored.

Administrative policy MAY require that the address not be included in the server's table of subnets. If so, then the notification is ignored.

Otherwise, the server MUST compare the subnet ID and subnet mask to those in each row in the table of connected subnets. If a matching row is not found, the server MUST create one.

The server MUST then add the local address to the list of server addresses in the row for its subnet.

The server SHOULD then increment its metadata version.<13>

Then the server MUST send a Hello message to announce the new address, as specified in [WS-Discovery] section 4.1.

When a Hello message is sent, the server MUST send the message to the subnet whose address list changed, implying that the **/s:Envelope/s:Body/d:Hello/d:XAddrs** element MUST contain the server addresses in that subnet from the table of connected subnets. The server SHOULD send the message only to that subnet, in which case it SHOULD NOT contain addresses in other subnets.<14>

If the server sends the message to multiple subnets, the **/s:Envelope/s:Body/d:Hello/d:XAddrs** element MUST contain the server addresses in each of those subnets.

3.1.7.2 Remove an IP Address

When an IP address is to be deleted, the server MUST check whether the address is a member of the address list in any row of the table of connected subnets. If not, the notification is ignored.

Otherwise, the server MUST remove the address from the list for that row. The server SHOULD then increment its metadata version.<15>

If the list contains other addresses, the server MUST send a Hello message to update the address list, as specified in [WS-Discovery] section 4.1. The server MAY send a Bye message to the subnet of the deleted address, using the current instance GUID for the **/s:Body/d:Bye/a:EndpointReference/a:Address** child element, as specified in [WS-Discovery] section 4.2.

When a Hello message is sent, the server MUST send the message to the subnet whose address list changed, implying that the **/s:Envelope/s:Body/d:Hello/d:XAddrs** element MUST contain the server addresses in that subnet from the table of connected subnets. The server SHOULD send the message only to that subnet, in which case it SHOULD NOT contain addresses in other subnets.<16>

Copyright © 2008 Microsoft Corporation.

If the server sends the Hello message to multiple connected subnets, the **/s:Envelope/s:Body/d:Hello/d:XAddrs** element MUST contain the server addresses in each of those subnets.

3.2 Client Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The client behaves as a WS-Discovery client, as used throughout [WS-Discovery].

3.2.1.1 Table of Subnets

The client maintains a table of all IP subnets for which it currently holds the following:

- A local IP address.
- A table of server addresses.

Each row of the table represents a single subnet and includes the following data:

- A flag M_ATTACHED that is true if the client is currently attached to this subnet.
- A pointer to a table of server addresses.
- For each IPv4 subnet, the client saves its subnet ID, subnet mask, and DNS domain suffix (or an empty string if not available).
- For each IPv6 subnet, the client saves its subnet ID and subnet mask.

3.2.1.2 Table of Servers

The client maintains a table of all servers that are present or cached. Each row of the table represents a single server and includes:

- Its fully qualified domain name (FQDN).
- The version of the BITS Peer-Caching: Content Retrieval Protocol that it supports, as specified in [MS-BPCR] sections 3.1.1.1 and 1.7.
- A Boolean that is true if the server has been authenticated by implementation-dependent means.<17>
- (OPTIONAL) A unique GUID representing the server instance. This datum is used only if the client chooses to process Bye messages.

3.2.1.3 Tables of Server Addresses

For each subnet, the client maintains a table of servers and their local addresses. Each row of the table represents a single server address and includes:

- Its fully qualified domain name.

Copyright © 2008 Microsoft Corporation.

- Its last known address in this subnet.
- The UTC time of the last time this address was refreshed.

The UTC time of the address is updated whenever a Hello or Probe-Match from the server is processed successfully and contains this address. A higher-layer protocol SHOULD also update the time when the server is successfully contacted through Content Retrieval.

3.2.1.4 Scope List

The client maintains a list of active WS-Discovery scopes defined by implementation-dependent means.<18>

3.2.2 Timers

3.2.2.1 Discovery Timer

A higher-layer protocol may signal the BITS Peer-Caching: Peer Discovery Protocol to discover more servers. Each such request is called a **discovery**. Each discovery has a finite lifetime after which the higher-layer protocol is notified that the discovery has completed. Each discovery contains a discovery timer to track the lifetime of the discovery. The default value is 30 seconds.

An implementation MAY modify the default to any positive number of seconds, but it SHOULD not set it to a value lower than the completion time of the SOAP-over-UDP transmission algorithm. WS-Discovery's use of SOAP-over-UDP is as specified in [WS-Discovery] section 2.4. For more information about SOAP-over-UDP, see [SOAP-UDP].

3.2.2.2 Discovery Suppression Timer

The protocol imposes a waiting period after sending a Probe message to avoid an inundation of network traffic from repeated discoveries. During this waiting period, new discovery requests from a higher-layer protocol complete immediately, without triggering a Probe message. The default value for this timer is 10 minutes; it may be any nonnegative value.

3.2.2.3 Address Scavenger Timer

To reduce accumulation of obsolete server and address entries, each address contains a scavenger timer. The default timer interval is seven days and MAY be any positive value.

3.2.3 Initialization

The client MUST identify its WS-Discovery scope(s) by implementation-dependent means.<19>

The client MUST begin listening for messages. Transport information is as specified in [WS-Discovery] section 2.4.

The client MUST enumerate the subnets to which it is attached and send itself an "attach to subnet" notification for each (see section 3.2.7.1 for details).

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Clear the Table of Servers

A higher-layer protocol MAY signal the BITS Peer-Caching: Peer Discovery Protocol to remove all servers from the table of servers.

Copyright © 2008 Microsoft Corporation.

When this occurs, the client MUST clear the table of servers. It first checks the M_ATTACHED flag in each row in the table of subnets. For rows where M_ATTACHED is false, the client MUST delete the associated table of server addresses and then delete the row. For the remaining rows, the client MUST clear the associated table of server addresses.

The client MUST mark the Discovery Suppression timer as not-pending, so that the next discovery request will not be suppressed.

3.2.4.2 Discovery Request

A higher-layer protocol MAY signal the protocol to discover more servers.

When this occurs, the client MUST check the Discovery Suppression timer. If it is still pending, then a discovery has completed recently and the client MUST immediately notify the higher layer that the discovery has terminated.

Otherwise, the client MUST send a Probe message.

3.2.4.3 Cancel Discovery Request

When a discovery request is canceled, the client MUST cancel the discovery's timer and immediately notify the higher layer that the discovery has terminated.

3.2.4.4 Enumerate Server Addresses

A higher-layer protocol MAY ask for a list of all online servers. When this occurs, the client MUST create a list of servers as follows:

The client creates an empty list of servers. It then enumerates the table of addresses for each subnet with M_ATTACHED == true. For each address, the client adds the server to the list if it is not already there, then it appends the address to the server entry.

The result is a list of all servers currently sharing at least one subnet with the client, together with the relevant server addresses. This list is returned to the higher-layer protocol.

3.2.4.5 Update Server Address Time Stamp

A higher-layer protocol MAY signal the client to update the time stamp of a particular server address. When this occurs, the client SHOULD set the time stamp of the address to the current UTC time and restart the address's scavenger timer.

3.2.4.6 Shut Down

When the protocol is shut down, the client:

- MUST stop listening for new messages from the transport. Queued messages MAY be processed.
- SHOULD close the network ports as specified in [WS-Discovery] section 2.4.<20>
- MUST cancel all active discovery requests, notifying the higher-layer protocol for each one.
- SHOULD save all tables in section 3.2.1 for use in the next instantiation.

Copyright © 2008 Microsoft Corporation.

3.2.5 Message Processing Events and Sequencing Rules

The client MUST verify that each received message matches the schema as specified in [WS-Discovery], Appendix III, discarding malformed messages. Further parsing depends on the message type.

3.2.5.1 Hello

The client adheres to the requirements as specified in [WS-Discovery] section 4.1. In addition, BITS Peer-Caching: Peer Discovery Protocol adds the following requirements.

The client MUST verify that the message satisfies the requirements in sections 2.2 and 2.2.1, discarding the message if not.

The client MUST verify that the message's **/s:Envelope/s:Body/d:Hello/d:Types** element includes the type **msbits:PeerServer**, discarding the message if not.

The client MAY verify other administratively defined criteria, discarding the message if they are not satisfied.<21>

The client SHOULD add the server data to its tables as follows:

1. *Find or create the server row.* From the **/s:Body/d:Hello/a:EndpointReference** element, set M_INSTANCE to the value of the **a:Address** child element, M_VERSION to the value of the **msbits:version** child element, and M_FQDN to the value of the **msbits:Fqdn** child element. Then, for each row in the table of servers, perform a case-insensitive comparison of the row's FQDN with M_FQDN. If a row matches, update the row's protocol version with M_VERSION and skip to step 2; otherwise, create a new row using M_FQDN, M_VERSION, and M_INSTANCE.

2. *Integrate server addresses.* For each address URI in the /**s:Body/d:Hello/d:XAddrs** element, convert into an IPv4 or an IPv6 address as appropriate, then check whether the address matches an existing row in the table of subnets. If no row matches, the address MUST be discarded. Otherwise, search in the associated table of server addresses for a row matching M_FQDN. If no row matches, create a new row by using the address from the message. Set the timestamp of the row to the current UTC time; set the address of the row to the current message address.

3.2.5.2 Bye

The client adheres to the requirements as specified in [WS-Discovery] section 4.2. In addition, (BITS) Peer-Caching: Peer Discovery Protocol adds the following requirements.

When a client receives a Bye message, it MAY discard the message.<22>

Otherwise, it MUST verify that the **/s:Body/d:Bye/a:EndpointReference/a:Address** element contains a valid instance GUID; if not, the message MUST be discarded. The client MUST search the table of servers for a row whose instance GUID matches the GUID in the element. If a matching row is found, the client MUST delete the associated table of server addresses and the matching row.

3.2.5.3 Probe-Match

The client adheres to the requirements as specified in [WS-Discovery] section 5. In addition, BITS Peer-Caching: Peer Discovery Protocol adds the following requirements.

The client MUST verify that the message satisfies the requirements in sections 2.2 and 2.2.4; if not, the message MUST be discarded.

Copyright © 2008 Microsoft Corporation.

The client MUST verify that the message's **/s:Envelope/s:Body/d:Probe-Match/d:Types** element includes the type **msbits:PeerServer**, discarding the message if not.

The client MUST verify that at least one of its scopes matches a scope in the message by using the rules as specified in [WS-Discovery] section 5.1, discarding the message if not.

The client SHOULD add the server data to the table of servers as described in 3.2.5.1, with references to the **/s:Envelope/s:Body/d:Hello** element replaced by **/s:Envelope/s:Body/d:Probe-Match.**

The client MAY enforce limits on the number or rate of Probe-Match messages processed.

3.2.5.4 Other Messages

Other messages MUST be ignored.

3.2.6 Timer Events

3.2.6.1 Discovery Time-Out

When a discovery's timer expires, the higher-layer protocol MUST be notified that the discovery has completed; then, the discovery MUST be deleted.

3.2.6.2 Discovery Suppression Time-Out

When the discovery suppression timer expires, nothing happens. However, the next discovery will trigger a Probe message because the client will see that the timer is expired.

3.2.6.3 Address Scavenger Time-Out

When a server's address scavenger timer expires, the address SHOULD be deleted from its table of server addresses. If its table is for a detached subnet and no other addresses remain, the table SHOULD be deleted. If no other addresses of that server remain in any subnet's table of addresses, the server SHOULD be deleted from the table of servers.

3.2.7 Other Local Events

3.2.7.1 Attach to a Subnet

When the client host attaches to an IP subnet, the client MUST check whether any existing row in the table of subnets matches the new subnet. If no row matches, the client MUST:

1. Create a new row containing the data for the new subnet.
2. Associate a new empty table of server addresses with the row.
3. Set the row's M_ATTACHED flag to true.

3.2.7.2 Detach from a Subnet

When the client host detaches from an IP subnet, the client MUST find the row with matching subnet information in the table of subnets. Then the client MUST:

1. Set the row's ATTACHED flag to false.

Copyright © 2008 Microsoft Corporation.

2. Check if the associated table of server addresses is empty. If so, the client MUST delete the associated table and the current row.

Copyright © 2008 Microsoft Corporation.

4 Protocol Examples

The following sections describe several operations used in common scenarios to illustrate the function of the BITS Peer-Caching: Peer Discovery Protocol.

4.1 Hello Message at Server Startup and Bye Message at Shutdown

A host named \\myclient is a member of the Windows domain MyDomain. The host is connected to a single network, holding both an IPv4 address and an IPv6 address. When the BITS Peer-Caching: Peer Discovery Protocol server is started, the host sends the following message:

```
(1)<?xml version="1.0" encoding="utf-8" ?>
(2)<soap:Envelope
(3)xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
(4)xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(5)xmlns:wsd="http://schemas.xmlsoap.org/ws/2005/04/discovery"
(6)xmlns:msbits=
"http://schemas.microsoft.com/windows/2005/05/BITS/cache" >
(7)<soap:Header>
(8)<wsa:To>
(9)urn:schemas-xmlsoap-org:ws:2005:04:discovery
(10)</wsa:To>
(11)<wsa:Action>
(12)http://schemas.xmlsoap.org/ws/2005/04/discovery/Hello
(13)</wsa:Action>
(14)<wsa:MessageID>
(15)urn:uuid:16d1ca53-23c0-4e27-accf-2bf71377f49e
(16)</wsa:MessageID>
(17)<wsd:AppSequence
(18)InstanceId="1169067015" MessageNumber="2" >
(19)</wsd:AppSequence>
(20)</soap:Header>
(21)<soap:Body>
(22)<wsd:Hello>
(23)<wsa:EndpointReference>
(24)<wsa:Address>
(25)uuid:A99558EB-C1D8-49D3-9476-8B9A6571800B
(26)</wsa:Address>
(27)<msbits:Fqdn>
(28)myclient.mydomain.com
(29)</msbits:Fqdn>
(30)<msbits:version>
(31)1
(32)</msbits:version>
(33)</wsa:EndpointReference>
(34)<wsd:Types>
(35)msbits:PeerServer
(36)</wsd:Types>
(37)<wsd:Scopes>
(38)http://mydomain.com
(39)</wsd:Scopes>
(40)<wsd:XAddrs>
(41)https://[2001:4898:2c:2:1db1:40d8:28fb:79d0]
(42)https://192.68.1.1
(43)</wsd:XAddrs>
(44)<wsd:MetadataVersion>
(45)1
```


Copyright © 2008 Microsoft Corporation.

```
(46)</wsd:MetadataVersion>
(47)</wsd:Hello>
(48)</soap:Body>
(49)</soap:Envelope>
```

The packet is sent four times: twice to the IPv4 address 239.255.255.250 port 3702, and twice to the IPv6 address FF02::C port 3702. This follows the algorithm in [SOAP-UDP] Appendix I.

When the BITS Peer-Caching: Peer Discovery Protocol server is shut down, the following Bye message is sent. Like the Hello message, it is sent twice to each multicast address.

```
(1)<?xml version="1.0" encoding="utf-8" ?>
(2)<soap:Envelope
(3)xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
(4)xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(5)xmlns:wsd="http://schemas.xmlsoap.org/ws/2005/04/discovery" >
(6)<soap:Header>
(7)<wsa:To>
(8)urn:schemas-xmlsoap-org:ws:2005:04:discovery
(9)</wsa:To>
(10)<wsa:Action>
(11)http://schemas.xmlsoap.org/ws/2005/04/discovery/Bye
(12)</wsa:Action>
(13)<wsa:MessageID>
(14)urn:uuid:e1c429f4-661d-4f98-a6d3-ce712efa28b7
(15)</wsa:MessageID>
(16)<wsd:AppSequence
(17)InstanceId="1169067015"
(18)MessageNumber="3" >
(19)</wsd:AppSequence>
(20)</soap:Header>
(21)<soap:Body>
(22)<wsd:Bye>
(23)<wsa:EndpointReference>
(24)<wsa:Address>
(25)uuid:A99558EB-C1D8-49D3-9476-8B9A6571800B
(26)</wsa:Address>
(27)</wsa:EndpointReference>
(28)</wsd:Bye>
(29)</soap:Body>
(30)</soap:Envelope>
```

4.2 Client Probe with Probe-Match Replies

A host named \\client1 is a member of the Windows domain MyDomain. The host is connected to a single network, holding both an IPv4 address and an IPv6 address.

When the BITS Peer-Caching: Peer Discovery Protocol client initiates a discovery, the host sends the following message. Line 51 indicates that the search is for Peer Discovery servers, and line 55 indicates the scope to match (here, a Windows domain).

```
(1)<?xml version="1.0" encoding="utf-8" ?>
(2)<soap:Envelope
```


Copyright © 2008 Microsoft Corporation.

```
(3)xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
(4)xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(5)xmlns:wsd="http://schemas.xmlsoap.org/ws/2005/04/discovery"
(6)xmlns:msbits="http://schemas.microsoft.com/windows/2005/05/BITS/cache" >
(7)<soap:Header>
(8)<wsa:To>
(9)urn:schemas-xmlsoap-org:ws:2005:04:discovery
(10)</wsa:To>
(11)<wsa:Action>
(12)http://schemas.xmlsoap.org/ws/2005/04/discovery/Probe
(13)</wsa:Action>
(14)<wsa:MessageID>
(15)urn:uuid:7895122d-f9d6-4cb9-b819-872f24c271b9
(16)</wsa:MessageID>
(17)</soap:Header>
(18)<soap:Body>
(19)<wsd:Probe>
(20)<wsd:Types>
(21)msbits:PeerServer
(22)</wsd:Types>
(23)<wsd:Scopes
(24)MatchBy="http://schemas.xmlsoap.org/ws/2005/04/discovery/rfc2396" >
(25)http://mydomain.com
(26)</wsd:Scopes>
(27)</wsd:Probe>
(28)</soap:Body>
(29)</soap:Envelope>
```

The message is sent four times: twice to the IPv4 address 239.255.255.250 port 3702, and twice to the IPv6 address FF02::C port 3702. This follows the algorithm in Appendix I of [SOAP-UDP].

Two Peer Discovery servers in the same Windows domain and one in a different domain receive the Probe.

The host \\peer1 is a member of the MyDomain domain; after 150 milliseconds it replies with the following message:

```
(1)<?xml version="1.0" encoding="utf-8" ?>
(2)<soap:Envelope
(3)xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
(4)xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(5)xmlns:wsd="http://schemas.xmlsoap.org/ws/2005/04/discovery"
(6)xmlns:msbits="http://schemas.microsoft.com/windows/2005/05/BITS/cache" >
(7)<soap:Header>
(8)<wsa:To>
(9)http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
(10)</wsa:To>
(11)<wsa:Action>
(12)http://schemas.xmlsoap.org/ws/2005/04/discovery/ProbeMatches
(13)</wsa:Action>
(14)<wsa:MessageID>
(15)urn:uuid:769dfff7-e778-4506-a764-0cbb26cb0f60
(16)</wsa:MessageID>
(17)<wsa:RelatesTo>
(18)urn:uuid:7895122d-f9d6-4cb9-b819-872f24c271b9
(19)</wsa:RelatesTo>
(20)<wsd:AppSequence
(21)InstanceId="1168888181"
(22)MessageNumber="434">
(23)</wsd:AppSequence>
(24)</soap:Header>
```


Copyright © 2008 Microsoft Corporation.

```
(25)<soap:Body>
(26)<wsd:ProbeMatches>
(27)<wsd:ProbeMatch>
(28)<wsa:EndpointReference>
(29)<wsa:Address>
(30)uuid:FDEFC35B-3B18-4E1C-B970-09F811D00304
(31)</wsa:Address>
(32)<msbits:Fqdn>
(33)peer1.mydomain.com
(34)</msbits:Fqdn>
(35)<msbits:version>
(36)1
(37)</msbits:version>
(38)</wsa:EndpointReference>
(39)<wsd:Types>
(40)msbits:PeerServer
(41)</wsd:Types>
(42)<wsd:Scopes>
(43)http://mydomain.com
(44)</wsd:Scopes>
(45)<wsd:XAddrs>
(46)https://[2001:4898:2c:2:dc2c:a67c:68ed:4c0b]
(47)https://192.168.1.20
(48)</wsd:XAddrs>
(49)<wsd:MetadataVersion>
(50)1
(51)</wsd:MetadataVersion>
(52)</wsd:ProbeMatch>
(53)</wsd:ProbeMatches>
(54)</soap:Body>
(55)</soap:Envelope>
```

The UUID in line 18 matches the one in line 15 of the Probe. Line 33 reflects the host's FQDN, line 43 reflects the host's Windows domain, and lines 46-47 contain the host's IP addresses.

The host \\peer2 is a member of the MyDomain domain; after 800 milliseconds it replies with the following message:

```
(1)<?xml version="1.0" encoding="utf-8" ?>
(2)<soap:Envelope
(3)xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
(4)xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
(5)xmlns:wsd="http://schemas.xmlsoap.org/ws/2005/04/discovery"
(6)xmlns:msbits="http://schemas.microsoft.com/windows/2005/05/BITS/cache" >
(7)<soap:Header>
(8)<wsa:To>
(9)http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
(10)</wsa:To>
(11)<wsa:Action>
(12)http://schemas.xmlsoap.org/ws/2005/04/discovery/ProbeMatches
(13)</wsa:Action>
(14)<wsa:MessageID>
(15)urn:uuid:a7eb1f92-cc38-434a-a619-bcb0f6a85de4
(16)</wsa:MessageID>
(17)<wsa:RelatesTo>
(18)urn:uuid:7895122d-f9d6-4cb9-b819-872f24c271b9
(19)</wsa:RelatesTo>
(20)<wsd:AppSequence
(21)InstanceId="1168539790"
(22)MessageNumber="666" >
(23)</wsd:AppSequence>
```


Copyright © 2008 Microsoft Corporation.

```
(24)</soap:Header>
(25)<soap:Body>
(26)<wsd:ProbeMatches>
(27)<wsd:ProbeMatch>
(28)<wsa:EndpointReference>
(29)<wsa:Address>
(30)uuid:80991AC9-6A0F-440D-9ACB-45D48F1A2D7F
(31)</wsa:Address>
(32)<msbits:Fqdn>
(33)peer2.mydomain.com
(34)</msbits:Fqdn>
(35)<msbits:version>
(36)1
(37)</msbits:version>
(38)</wsa:EndpointReference>
(39)<wsd:Types>
(40)msbits:PeerServer
(41)</wsd:Types>
(42)<wsd:Scopes>
(43)http://mydomain.com
(44)</wsd:Scopes>
(45)<wsd:XAddrs>
(46)https://[2001:4898:2c:2:2cc8:dae1:1bfb:4aea]
(47)https://192.168.1.21
(48)</wsd:XAddrs>
(49)<wsd:MetadataVersion>
(50)1
(51)</wsd:MetadataVersion>
(52)</wsd:ProbeMatch>
(53)</wsd:ProbeMatches>
(54)</soap:Body>
(55)</soap:Envelope>
```

The host \\products is a member of a different Windows domain. Line 25 of the Probe message fails to match the host's own WS-Discovery scope, and so the Probe is discarded without response.

Copyright © 2008 Microsoft Corporation.

5 Security

The following sections specify security considerations for implementers of the BITS Peer-Caching: Peer Discovery Protocol.

5.1 Security Considerations for Implementers

5.1.1 Potential for High Unicast Traffic

WS-Discovery does not provide a way to control the number or pace of replies to a Probe message. In a very large network, a client may be overwhelmed by many server replies.

5.1.2 Lack of Message Authentication

This protocol does not provide any authentication for messages. It is possible for a malicious host to send incorrect Hello, Bye, and Probe-Match messages in order to confuse a client. A client MUST consider all information gained from this protocol as insecure until corroborated by other means.

5.2 Index of Security Parameters

This protocol does not define any security parameters.

Copyright © 2008 Microsoft Corporation.

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows Vista
- Windows Server 2008

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

<1> Section 2.2.1: Windows always sends a single scope, defined by prepending the string "https://" to the FQDN of the host.

<2> Section 2.2.1: Windows includes one **XAddr** for each active non-loopback IPv4 address and each globally aggregatable IPv6 address.

<3> Section 2.2.3: Windows always sends a single scope, defined by prepending the string "https://" to the FQDN of the host.

<4> Section 2.2.4: Windows always sends a single scope, defined by prepending the string "https://" to the FQDN of the host.

<5> Section 2.2.4: Windows includes one **XAddr** for each active non-loopback IPv4 address and each globally aggregatable IPv6 address.

<6> Section 2.2.5: Windows does not send these messages, and ignores them if received.

<7> Section 3.1.3: Windows resets the metadata version to zero each time the protocol is initialized.

<8> Section 3.1.3: Windows defines a single scope by prepending the string "https://" to the FQDN of the host.

<9> Section 3.1.4.1: Windows sends a Bye message.

<10> Section 3.1.4.1: Windows closes the ports if and only if no other WS-Discovery–based protocols are active on the host.

<11> Section 3.1.5.1: Windows sends a URI list containing all server addresses from the table of connected subnets.

<12> Section 3.1.5.2: Windows ignores Resolve messages.

<13> Section 3.1.7.1: Windows does not increment its metadata version when the address list changes.

<14> Section 3.1.7.1: Windows sends a URI list containing all server addresses from the table of connected subnets.

<15> Section 3.1.7.2: Windows does not increment its metadata version when the address list changes.

<16> Section 3.1.7.2: Windows sends a URI list containing all server addresses from the table of connected subnets.

Copyright © 2008 Microsoft Corporation.

<17> Section 3.2.1.2: Windows authenticates servers by using the BITS Peer-Caching: Peer Authentication Protocol, as specified in [MS-BPAU].

<18> Section 3.2.1.4: Windows defines a single scope by prepending the string "https://" to the FQDN of the host.

<19> Section 3.2.3: Windows defines a single scope by prepending the string "https://" to the FQDN of the host.

<20> Section 3.2.4.6: Windows closes the ports if and only if no other WS-Discovery–based protocols are active on the host.

<21> Section 3.2.5.1: Windows verifies that the **/s:Body/d:Hello/a:EndpointReference/msbits:Fqdn** element matches the Windows domain to which the client belongs. Windows supports the protocol only on hosts that are members of a Windows domain.

<22> Section 3.2.5.2: Windows always discards Bye messages.

Copyright © 2008 Microsoft Corporation.

7 Index

Copyright © 2008 Microsoft Corporation.

N

O

P

R

S

T

U

V

W

Copyright © 2008 Microsoft Corporation.