[MS-PCCRD]:
Peer Content Caching and Retrieval Discovery Protocol Specification

---

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This 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 technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** 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.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of 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.

Tools. The Open Specifications do 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. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
12/05/2008	0.1	Major	Initial Availability
01/16/2009	0.1.1	Editorial	Revised and edited the technical content.
02/27/2009	0.2	Minor	Updated the technical content.
04/10/2009	0.3	Minor	Updated the technical content.
05/22/2009	0.4	Minor	Updated the technical content.
07/02/2009	1.0	Major	Updated and revised the technical content.
08/14/2009	1.0.1	Editorial	Revised and edited the technical content.
09/25/2009	1.1	Minor	Updated the technical content.
11/06/2009	2.0	Major	Updated and revised the technical content.
12/18/2009	2.0.1	Editorial	Revised and edited the technical content.
01/29/2010	2.1	Minor	Updated the technical content.
03/12/2010	3.0	Major	Updated and revised the technical content.
04/23/2010	3.1	Minor	Updated the technical content.
06/04/2010	4.0	Major	Updated and revised the technical content.
07/16/2010	4.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	4.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	4.0	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	4.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	4.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	4.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This document specifies the Discovery Protocol for the Windows Peer Content Caching and Retrieval framework.

The Windows Peer Content Caching and Retrieval framework is a **content** caching and retrieval framework based on a **peer**-to-peer discovery and distribution model. It is designed to reduce bandwidth consumption on branch-office wide-area-network (WAN) links by having **clients** retrieve content from distributed caches, when distributed caches are available, rather than from the **content servers**, which are often located remotely from branch offices over the WAN links. The peers themselves act as caches from which they serve other requesting peers. The main benefit is to reduce operation costs by reducing WAN link utilization, while providing faster downloads from the **local area network (LAN)** in the branch office.

The Discovery Protocol in the framework is utilized by peers adopting the client role to discover content among other peers. It is based on the Web Service Dynamic Discovery Protocol (also called WSD) [WS-Discovery] standard, with no changes in its protocol syntax and behaviors. Therefore, this document refers to [WS-Discovery] for the detailed specification but provides the product-specific message formats and describes how the Discovery Protocol fits into the overall framework.

1.1 Glossary

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

globally unique identifier (GUID)
local area network (LAN)
multicast
SOAP
URI
UTC (Coordinated Universal Time)
Web Services Description Language (WSDL)
WSDL message
WSDL operation
XML
XML namespace
XML schema (XSD)

The following terms are specific to this document:

block: One of the fixed-size chunks of **content** that compose a **segment**. Each **segment** is divided into one or more **blocks** of equal size (64 kilobytes) except for the last **block**, which in general can be smaller. Every **block** belongs to a specific **segment**, and within a **segment**, **blocks** are identified by their progressive index. (Block 0 is the first **block** in the **segment**, block 1 is the second, and so on.) Given the fixed size of **blocks**, their index in the **segment** also indicates their actual byte-offset in the **segment** itself. (See [MS-PCCRC] for more details.)

client: Within this protocol document, a **peer** that is configured to access certain **content**. It is acting as a WSD **client** in the Discovery protocol.

content: Items that correspond to a file that an application attempts to access. Examples of **content** include Web pages and documents stored on either HTTP servers or SMB file servers. Each **content** item consists of an ordered collection of one or more **segments** of the same

size (32 megabytes), although they can be shorter if they represent the last **segment** of a **content** item the size of which is not a multiple of the **segment** size.

content server: The original server that a **peer** contacts to obtain either the hashes of the **content** (see [MS-PCCRC]) or the actual **content** when it is not available from the **peers**.

peer: Within this protocol document, local nodes participating in the **content** caching and retrieval framework. A **peer** is a node that both accesses the **content** and serves the **content** it caches for other **peers**.

Probe: The WSD protocol message sent by a **client** to discover certain **content**.

ProbeMatch: The WSD protocol message sent by a server peer to the requesting **client** when it has the **content** the **client** is looking for.

segment: A unit of **content** for discovery purposes. A **segment** is identified on the network by its public identifier, also known as **segment ID** or **HoHoDk**. A **segment** does not belong to any particular **content**; it can be shared by many **content** items if all those **content** items have an identical **segment**-sized portion at some offset.

segment ID/HoHoDk: A hash that represents the **content**-specific label or public identifier that is used to discover **content** from other **peers**. This identifier is disclosed freely in WSD **Probe multicast** messages. Knowledge of the identifiers does not prove authorization to access the actual **content**.

server peer: A **peer** that has the **content** in its local cache and that will serve the **content** to other **peers** requesting it. A **server peer** is acting as a WSD server or service target in the Discovery Protocol.

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-PCCRC] Microsoft Corporation, "Peer Content Caching and Retrieval: Content Identification", December 2008.

[MS-PCCRR] Microsoft Corporation, "Peer Content Caching and Retrieval: Retrieval Protocol Specification", December 2008.

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980, http://www.ietf.org/rfc/rfc768.txt

[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

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, http://www.ietf.org/rfc/rfc3986.txt

[SOAP-UDP] Combs, H., Justice, M., Kakivaya, G., 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.

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

[WS-Discovery] Beatty, J., Kakivaya, G., Kemp D., et al., "Web Services Dynamic Discovery (WS-Discovery)", April 2005, http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf

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

[WSDL] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1", W3C Note, March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[XMLNS-3ED] World Wide Web Consortium, "Namespaces in XML 1.0 (Third Edition)", December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA1.1/2] Peterson, D., Ed., Biron, P.V., Ed., Malhotra, A., Ed., and Sperberg-McQueen, C.M., Ed., "XML Schema 1.1 Part 2: Datatypes", W3C Working Draft, June 2008, http://www.w3.org/TR/2008/WD-xmlschema11-2-20080620/

1.2.2 Informative References

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

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

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

1.3 Overview

The Peer Content Caching and Retrieval Discovery Protocol is based on the Web Services Dynamic Discovery (WS-Discovery) protocol, which uses multicast to discover and locate services over the network. There are two modes of operations in WSD: client-initiated **Probes** and service-initiated announcements; both are sent through IP multicast to a predefined group. The Peer Content Caching and Retrieval Discovery Protocol uses the client-initiated Probes to query peer nodes for content.

In the Peer Content Caching and Retrieval Discovery Protocol, the peers looking for content are the WSD clients, sending out multicast WSD Probe messages with the hashes of the content. The peers that are serving the content take the server role of WSD, listening to Probes and replying to the querying clients with unicast **ProbeMatch** messages if the content hashes match the ones that are cached locally. To avoid the situation where multiple **server peers** overwhelm a client with ProbeMatch messages, per the WS-Discovery protocol, each server peer must wait for a backoff timer to expire before sending the ProbeMatch. The value of the backoff timer is set randomly between 1 millisecond and a maximum backoff time known as **APP_MAX_DELAY**.

1.4 Relationship to Other Protocols

The Discovery Protocol uses the Web Services Dynamic Discovery (WS-Discovery) protocol, which uses **SOAP**-over-UDP [SOAP-UDP] [RFC768] as its network transport. In the Windows Peer Content Caching and Retrieval framework, the Discovery Protocol is used by the Peer Content Caching and

Retrieval: Retrieval Protocol [MS-PCCRR] to discover other peers that have a particular **segment** of content.

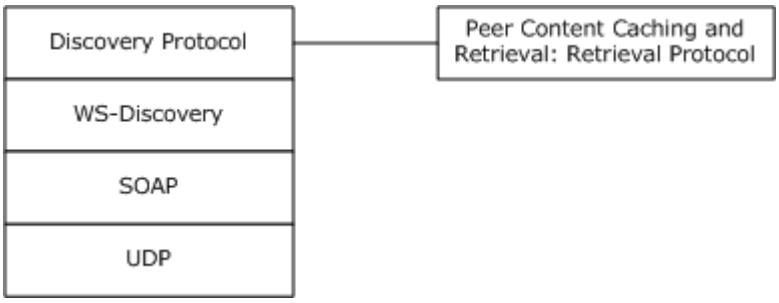

Figure 1: Layering of the protocol stack

1.5 Prerequisites/Preconditions

The Discovery Protocol depends on the Web Services Dynamic Discovery (WS-Discovery) protocol. The client requesting the discovery needs to have the content information, defined in the Content Information Specification [MS-PCCRC], for the content that it is looking for, and the peers answering the request need to have a (possibly empty) cache of content that they can check to see if they have the requested content stored locally in order to answer a query.

1.6 Applicability Statement

A client uses the Discovery Protocol to locate peers with the content it needs, in the context of the Peer Content Caching and Retrieval Framework. The protocol is thus appropriate (only) in settings where multiple peers and at least one content server support this framework.<1> It is particularly applicable in cases where the content server is available only over a relatively high-latency and/or low-bandwidth connection whereas the peers are available over relatively higher-bandwidth, lower-latency ones. In that case, the Framework, including this Discovery Protocol, offers the possibility of significant performance benefits.

The Discovery Protocol is used to find peers with the hashes listed in the Probe messages. It is not for discovering hashes for given names or URLs. It is necessary for the clients to contact the original content servers (as opposed to the peer caches) to obtain the Content Information data structure (see [MS-PCCRC]) containing hashes corresponding to the contents.

1.7 Versioning and Capability Negotiation

The Discovery Protocol is based on the WSD standard and does not define any new mechanisms for the following areas:

- Supported transports
- Protocol versions
- Security and authentication methods
- Localization
- Capability negotiation

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The Discovery Protocol uses the Web Services Dynamic Discovery (WS-Discovery) protocol, and the actual transport protocol is abstracted by WSD. The Discovery Protocol simply uses WSD with certain elements set to custom values and relies on WSD for the actual transport of the Receive Probe (section 3.2.4.1) and Receive ProbeMatch (section 3.1.4.1) messages.

2.2 Common Message Syntax

This section contains common definitions used by this protocol. The syntax of the definitions uses **XML schema** as defined in [XMLSCHEMA1.1/2], and **Web Services Description Language (WSDL)** as defined in [WSDL].

2.2.1 Namespaces

This specification defines and references various **XML namespaces** using the mechanisms specified in [XMLNS-3ED]. Although this specification associates a specific XML namespace prefix for each XML namespace that is used, the choice of any particular XML namespace prefix is implementation-specific and is not significant for interoperability.

Because the Discovery Protocol uses the WSD protocol [WS-Discovery], the namespace requirement will be a subset of the WSD namespaces.

Prefix	Namespace URI	Reference
soap	http://www.w3.org/2003/05/soap-envelope	[SOAP1.2-1/2003]
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing	[WSAddressing]
wsd	http://schemas.xmlsoap.org/ws/2005/04/discovery	[WS-Discovery]
PeerDist	http://schemas.microsoft.com/p2p/2007/09/PeerDistributionDiscovery	[MS-PCCRD]

The XML namespace **URI** that MUST be used by the implementation of the Discovery Protocol is:

```
http://schemas.microsoft.com/p2p/2007/09/PeerDistributionDiscovery
```

Note The above URI does not contain any schema definition. It is used merely as a tag to identify the namespace. All elements are defined in this document.

2.2.2 Messages

The following table summarizes the set of common **WSDL messages** defined by this specification. WSDL message definitions that are specific to a particular operation are described with the operation.

Message	Description
Probe	Sent by the endpoint to find the services defined by <Types> and <Scopes>.
ProbeMatch	Replied by the endpoint that can provide the services defined by <Types> and <Scopes>.

2.2.2.1 Probe

The Probe [WS-Discovery] message is part of the WS-Discovery standard. A client sends a Probe message to find a specific service defined by Types (section 2.2.3.1) and Scopes (section 2.2.3.2). The Discovery Protocol uses a WSD Probe with the <Types> and <Scopes> elements in the WSD message.

```
<wsd:Probe>
  <wsd:Types>
    PeerDist:PeerDistData
  </wsd:Types>
  <wsd:Scopes MatchBy="http://schemas.xmlsoap.org/ws/2005/04/discovery/strcmp0">
020000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
  </wsd:Scopes>
</wsd:Probe>
```

Types: The <Types> element, as specified in section 2.2.3.1.

Scopes: The <Scopes> element, as specified in section 2.2.3.2.

2.2.2.2 ProbeMatch

A ProbeMatch message SHOULD be sent in response to a Probe message (section 2.2.2.1) if the replying peer has the content segment listed in the Probe message as specified in section 3.2.4.1.

```
<wsd:ProbeMatches>
  <wsd:ProbeMatch>
    <wsa:EndpointReference>
      <wsa:Address>
              urn:uuid:87A89944-0230-43a5-AC4E-FAB1386C8E2C
      </wsa:Address>
    </wsa:EndpointReference>
    <wsd:Types>
      PeerDist:PeerDistData
    </wsd:Types>
    <wsd:Scopes>
020000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
    </wsd:Scopes>
    <wsd:XAddrs>
      157.59.141.183:54321
    </wsd:XAddrs>
    <wsd:MetadataVersion>
      1
    </wsd:MetadataVersion>
    <PeerDist:PeerDistData>
      <PeerDist:BlockCount>
        0000002A
      </PeerDist:BlockCount>
    </PeerDist:PeerDistData>
  </wsd:ProbeMatch>
</wsd:ProbeMatches>
```

Types: The <Types> element, as specified in section 2.2.3.1.

Scopes: The <Scopes> element, as specified in section 2.2.3.2.

Address: The <Address> element, as specified in section 2.2.3.3.

MetadataVersion: The <MetadataVersion> element, as specified in section 2.2.3.4.

InstanceId: The <InstanceId> element, as specified in section 2.2.3.5.

MessageNumber: The <MessageNumber> element, as specified in section 2.2.3.6.

XAddrs: The <XAddrs> element, as specified in section 2.2.3.7.

Any: The <Any> element, as specified in section 2.2.3.8.

2.2.3 Elements

The following table summarizes the set of common XML schema elements defined by this specification. XML schema element definitions that are specific to a particular message are described with the message.

Element	Description
<Types>	Represents the list of discovery provider types
<Scopes>	Represents the list of discovery provider scopes.
<Address>	Identifies the responding device.
<MetadataVersion>	Identifies the current metadata version.
<InstanceId>	Identifies the current instance of the device being published.
<MessageNumber>	Identifies the counter within the scope of the instance identifier for the current message.
<XAddrs>	Contains the list of transport URIs supported by the peer responding to the Probe message (with a ProbeMatch message).
<Any>	Used to indicate how many **blocks** in each discovered segment are available.

2.2.3.1 Types

The <Types> element represents the list of discovery provider types. When specified in the request message, the value indicates the Types to be searched for. When specified in the response, the value provides the Types that were found. The value of the element is set to indicate that the Probe message was sent by the Windows Peer Content Caching and Retrieval framework so that it can be checked on the receiving side to confirm this. This element MUST be set to:

```
PeerDist:PeerDistData
```

2.2.3.2 Scopes

The <Scopes> element represents the list of discovery provider scopes, where each element in the list is a string. When specified in the request message, the peer populates this value to indicate the segments it is searching for. When sending the response, the value provides the list of scopes present in the cache.

Each element of the list is actually a UTF-8-encoded string representation of the **HoHoDk** value in hexBinary format [XMLSCHEMA1.1/2] and represents one segment. (For details of HoHoDk, refer to [MS-PCCRC].) The Discovery Protocol uses the case-sensitive string comparison rule defined as part of the WSD standard schemas. Refer to [WS-Discovery] section 5.1 for details.

2.2.3.3 Address

The <Address> element identifies the responding device. It MUST be set to a **globally unique identifier (GUID)** as a "uuid:" scheme URI.<2> Refer to [WS-Discovery] for the detailed definitions of Address and the "uuid:" scheme URI. The following line shows an example of the Address identifier.

```
urn:uuid:87A89944-0230-43a5-AC4E-FAB1386C8E2C
```

2.2.3.4 MetadataVersion

The <MetadataVersion> element is the current metadata version and MUST be set to 1.

2.2.3.5 InstanceId

The <InstanceId> element is the identifier for the current instance of the device being published. It MUST be generated at run time when the service starts and MUST be incremented whenever the service is restarted. The recommendations on the initial value and processing rules of the InstanceId are specified in [WS-Discovery].<3>

2.2.3.6 MessageNumber

The <MessageNumber> element is the counter within the scope of the instance identifier for the current message. This MUST be generated at run time, and the MessageNumber MUST be incremented for each message. The MessageNumber of the first message sent within the scope of the current InstanceID SHOULD be "1". Note that the InstanceID and the use and increment of the same MessageNumber apply to other message types defined in [WS-Discovery] that the applications using the Discovery Protocol may send, not just the ProbeMatch message defined in the Discovery Protocol. The rules on initializing, processing, and incrementing the MessageNumber are specified in [WS-Discovery].

2.2.3.7 XAddrs

The <XAddrs> element contains the list of transport URIs supported by the peer responding to the Probe message (with a ProbeMatch message). Each transport URI string MUST contain an address and port number that can be used for connection by a remote host. For detailed format specifications on transport URIs containing IP addresses and port numbers, see [RFC3986].

2.2.3.8 Any

A custom XML section MUST be embedded in the <Any> element of the WSD message body. This field is used to indicate how many blocks in each discovered segment are available. The XML MUST be formatted as follows:

```
<PeerDist:PeerDistData>
   <PeerDist:BlockCount>
      List of block counts
   </PeerDist:BlockCount>
```

```
</PeerDist:PeerDistData>
```

PeerDistData: This element encloses all the custom metadata for Windows Branch Cache responses.

BlockCount: This element is a child of the <PeerDistData> element. This element is set to the UTF-8-encoded string representation of the hexBinary [XMLSCHEMA1.1/2] packed array of integers in network byte order. Each integer represents a block count. For example, if there are three block counts with the decimal values 25, 4, and 16, the hexBinary value of these block counts is 001900040010, corresponding to the three segments discovered.

2.2.4 Complex Types

This specification does not define any common XML Schema complex type definitions.

2.2.5 Simple Types

This specification does not define any common XML Schema simple type definitions.

2.2.6 Attributes

This specification does not define any common XML Schema attribute definitions.

2.2.7 Groups

This specification does not define any common XML Schema group definitions.

2.2.8 Attribute Groups

This specification does not define any common XML Schema attribute group definitions.

3 Protocol Details

The Discovery Protocol utilizes the Probe and ProbeMatch messages from the WSD Protocol. Refer to [WS-Discovery] for the detailed protocol operation. This section describes how the Probe and ProbeMatch messages are used in the Discovery Protocol.

3.1 Client Details

3.1.1 Abstract Data Model

The following is a list of the abstract data model elements for the protocol state on the client or the requesting peer:

HoHoDk List: A list of the segment IDs/HoHoDks of the segments that the peer is searching for.

Outstanding Probe List: A list of currently outstanding Probes, together with the HoHoDks contained in them.

3.1.2 Timers

The client peer MUST set a request timer when sending a Probe message to wait for a certain period of time for the incoming replies. The request timer SHOULD be configurable such that system administrators can adjust the value based on their current network environments. The default value of the request timer SHOULD be set within the range of 200 milliseconds to 500 milliseconds, and under all circumstances, the waiting period SHOULD NOT be smaller than the maximum backoff timer value as defined in section 3.2.2 to ensure that enough replies are received by a requesting peer. When the request timer expires on a requesting peer without receiving any ProbeMatch replies, the implementation MUST remove that Probe message and its associated **HoHoDk List** from the **Outstanding Probe List**.

3.1.3 Initialization

The elements for the Probe message MUST be initialized according to their corresponding values or rules, as specified in section 2.2.2.1.

The **HoHoDk List** (or segment ID list) on the requesting peer MUST be initialized with values supplied by applications using the Discovery Protocol. This list consists of the HoHoDks passed by the applications through the interface of the Discovery Protocol as specified in section 1.4.

3.1.4 Message Processing Events and Sequencing Rules

The following table summarizes the list of client **WSDL operations** as defined by this specification.

Operation	Description
Receive ProbeMatch	Triggered by the WSD layer on receiving a response to a Probe message.

3.1.4.1 Receive ProbeMatch

This event is triggered by the WSD layer on receiving a response to a Probe as specified in section 3.2.4.1. The peer that receives a ProbeMatch message MUST check the <Types> and <XAddrs> elements to verify that the response was sent by a peer on the local subnet, and it MUST check the HoHoDks to verify that at least one is associated with a Probe message on the **Outstanding Probe List**. If so, it MUST report the IP address and the port number from the <XAddrs> element of the

ProbeMatch message and the relevant HoHoDk from the <Scopes> element of the ProbeMatch message to the higher layer.

All malformed ProbeMatch messages MUST be silently discarded.

3.1.5 Timer Events

When the request timer for a specific Probe message expires on the requesting peer, that Probe message and its associated **HoHoDk List** are removed from the **Outstanding Probe List**.

3.1.6 Other Local Events

All malformed messages destined for the Windows Peer Content Caching and Retrieval framework MUST be silently discarded. All WSD messages destined for other services will be dispatched by the WSD service to other registered components for the specific service types. For general WSD message processing and error handling, refer to [WS-Discovery].

3.1.6.1 Discovery Probe Started

To start a Discovery Probe, the higher layer passes one or more segment HoHoDks from the content it is looking for to the client-role peer, which sends out a WSD Probe message that is modified, as specified in section 2.2.2.1. It then adds the Probe together with its HoHoDks to the **Outstanding Probe List** and sets the request timer for this Probe.

3.2 Server Details

3.2.1 Abstract Data Model

The following abstract data model is needed on the server peer:

Available Segment List: A list of all HoHoDks for all available segments, and their corresponding block counts.

3.2.2 Timers

The server peer MUST set a backoff timer (defined and explained in [WS-Discovery]) randomly between 1 millisecond and a maximum value (defined by the WSD Protocol constant **APP_MAX_DELAY**). If the server peer has the requested segment ID in the request, the server MUST send the Probe-Match message back to the requesting peer when the backoff timer expires. For a detailed description of the backoff timer, see [WS-Discovery]. This timer SHOULD be configurable such that system administrators can adjust the value based on their current network environments. By default, **APP_MAX_DELAY** SHOULD be set to 65 milliseconds.<4>

3.2.3 Initialization

The server peers MUST initialize an empty **Available Segment List** or populate the list with the HoHoDks supplied by the higher-layer applications of the Discovery Protocol. The server peers MUST also generate and set the initial values for Address (section 2.2.3.3), InstanceId (section 2.2.3.5), and MessageNumber (section 2.2.3.6).

3.2.4 Message Processing Events and Sequencing Rules

The following table summarizes the list of server WSDL operations as defined by this specification.

Operation	Description
Receive Probe	Triggered by the WSD layer on receiving a Probe message.

3.2.4.1 Receive Probe

This event is triggered by the WSD layer on receiving a Probe message. The event processing involves checking the <Types> and <Scopes> elements to verify that the Probe was initiated by a peer. The peer then SHOULD respond to the Probe if one or more HoHoDks in the Probe match any that are stored locally.<5>

If none of the HoHoDks stored locally match the ones in the Probe message, the Probe message MUST be silently discarded.

If the peer responds to the Probe, it MUST first retrieve the corresponding block counts for the stored segments whose HoHoDks match the ones in the Probe message. It MUST then construct and send the ProbeMatch message (as defined in section 2.2.2.2) back to the requesting peer, using the correct block counts.

All malformed Probe messages MUST be silently discarded.

3.2.5 Timer Events

The behavior of the backoff timer is described in [WS-Discovery].

3.2.6 Other Local Events

All malformed messages destined for the Windows Peer Content Caching and Retrieval framework MUST be silently discarded. All WSD messages destined for other services will be dispatched by the WSD service to other registered components for the specific service types. For general WSD message processing and error handling, refer to [WS-Discovery].

3.2.6.1 Segment ID Added or Removed

When the higher-layer protocol or applications of the Discovery Protocol add or remove a segment ID/HoHoDk, the server peer MUST update its **Available Segment List** accordingly, with the updated segment IDs supplied by the higher-layer protocol or application.

4 Protocol Examples

A client node in a branch office contacts a Peer Content Caching and Retrieval–capable content server to retrieve a Web page, http://vortex.example.com/sample.html. It receives a list of hashes that represent the indices of the content. The client then sends the following WSD Probe message.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:wsd="http://schemas.xmlsoap.org/ws/2005/04/discovery"
  xmlns:PeerDist=
"http://schemas.microsoft.com/p2p/2007/09/PeerDistributionDiscovery">
  <soap:Header>
    <wsa:To>
      urn:schemas-xmlsoap-org:ws:2005:04:discovery
    </wsa:To>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2005/04/discovery/Probe
    </wsa:Action>
    <wsa:MessageID>
      urn:uuid:91528b47-b96d-4e30-981f-308c0586926f
    </wsa:MessageID>
  </soap:Header>
  <soap:Body>
    <wsd:Probe>
      <wsd:Types>
        PeerDist:PeerDistData
      </wsd:Types>
      <wsd:Scopes
                MatchBy=
        "http://schemas.xmlsoap.org/ws/2005/04/discovery/strcmp0">
020000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
      </wsd:Scopes>
    </wsd:Probe>
  </soap:Body>
</soap:Envelope>
```

The <Types> and <Scopes> elements contain the Discovery Protocol–specific values. The <Scopes> element in this example shows only one HoHoDk.

Other peer nodes that receive this Probe message check whether the HoHoDks in the Probe identify a segment at least some blocks of which are available locally. If a match is found, the node sets a backoff timer and replies with the following ProbeMatch message after the timer expires.

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap=http://www.w3.org/2003/05/soap-envelope
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsd="http://schemas.xmlsoap.org/ws/2005/04/discovery"
    xmlns:PeerDist=
      "http://schemas.microsoft.com/p2p/2007/09/PeerDistributionDiscovery
    <soap:Header>
      <wsa:To>
     http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      </wsa:To>
      <wsa:Action>
```

```
      http://schemas.xmlsoap.org/ws/2005/04/discovery/ProbeMatches
      </wsa:Action>
      <wsa:MessageID>
        urn:uuid:7f4b09a3-96c9-4170-9eb3-1c805a52f444
      </wsa:MessageID>
      <wsa:RelatesTo>
        urn:uuid:91528b47-b96d-4e30-981f-308c0586926f
      </wsa:RelatesTo>
      <wsd:AppSequence InstanceId="1218765447" MessageNumber="4"/>
    </soap:Header>
    <soap:Body>
      <wsd:ProbeMatches>
        <wsd:ProbeMatch>
          <wsa:EndpointReference>
            <wsa:Address>
              urn:uuid:87A89944-0230-43a5-AC4E-FAB1386C8E2C
            </wsa:Address>
          </wsa:EndpointReference>
          <wsd:Types>
            PeerDist:PeerDistData
          </wsd:Types>
          <wsd:Scopes>
02000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
          </wsd:Scopes>
          <wsd:XAddrs>
            157.59.141.183:54321
          </wsd:XAddrs>
          <wsd:MetadataVersion>
            1
          </wsd:MetadataVersion>
          <PeerDist:PeerDistData>
            <PeerDist:BlockCount>
              0000002A
            </PeerDist:BlockCount>
          </PeerDist:PeerDistData>
        </wsd:ProbeMatch>
      </wsd:ProbeMatches>
    </soap:Body>
  </soap:Envelope>
```

The <Types>, <Scopes>, <XAddrs>, and <MetadataVersion> elements contain the appropriate Discovery Protocol–specific values. The <PeerDist:PeerDistData> block of syntax after the <MetadataVersion> element contains the block counts for the segment referred to in the <Scopes> element.

5 Security

The WSD messages are multicast on the local subnet. Because the Discovery Protocol works on top of the Web Services Dynamic Discovery (WS-Discovery) protocol, it leverages the security model of WSD. For details of the security model of WSD, refer to [WS-Discovery]. In addition, the Discovery Protocol rejects responses from any subnet other than the local subnet.

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Full WSDL

This protocol follows the Web Services Dynamic Discovery (WS-Discovery) protocol. Its Web Services Description Language [WSDL] is identical to that of WS-Discovery and can be found in [WS-Discovery].

7 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Windows Vista® operating system
- Windows Server® 2008 operating system
- Windows® 7 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 1.6: For Windows Vista and Windows Server 2008, support for the client-side elements of this protocol is available only via the optional installation of the Background Intelligent Transfer Service by way of the Windows Management Framework. Support for the server-side elements of this protocol is not available for Windows Vista or Windows Server 2008.

<2> Section 2.2.3.3: The Windows implementation generates a new GUID for the URI when the service starts.

<3> Section 2.2.3.5: In Windows 7 and Windows Server 2008 R2, the InstanceId is generated by taking the boot time of the service in seconds since 1970-01-01 Time 00:00:00 **UTC**.

<4> Section 3.2.2: In Windows 7 and Windows Server 2008 R2, the default value for the request timer is 300 milliseconds, and the default maximum backoff timer value on the server peer is 65 milliseconds. However, these two values are configurable in Windows 7 and Windows Server 2008 R2 and can be set by system administrators based on the specific network environment.

<5> Section 3.2.4.1: In Windows 7 and Windows Server 2008 R2, the following rules are used to determine whether a server peer will reply to a Probe message when it has the requested segment.

When the server peer first requests and downloads a segment as a client, it records the number of replies it receives and whether the replies contain the entire segment or just a portion of the segment. Later, when the server peer receives a Probe message for that segment, it proceeds as follows (where n is an administratively configurable parameter with the default value n = 10):

- If there were fewer than n replies, the server peer will respond to the new Probe after the backoff timer expires.
- If there were at least n replies:
  - If there were at least n full-segment replies, the server peer will not reply to the new Probe.
  - If there were fewer than n full-segment replies, the server peer will randomly determine whether to reply to the new Probe, with a probability of 0.33 for replying.

8 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

9 Index