[MS-PCHC]:
Peer Content Caching and Retrieval:
Hosted Cache Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
07/16/2010	1.0	New	First Release.
08/27/2010	1.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	1.0.1	Editorial	Changed language and formatting in the technical content.
11/19/2010	1.0.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	1.0.1	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	1.0.1	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	1.0.1	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	1.0.1	No change	No changes to the meaning, language, or formatting of the technical content.
06/17/2011	1.1	Minor	Clarified the meaning of the technical content.

Contents

1 Introduction

This is a specification of the Peer Content Caching and Retrieval: Hosted Cache Protocol. It is used by clients to offer metadata to a hosted cache server.

1.1 Glossary

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

Domain Name System (DNS)
Generic Security Services (GSS)
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
Simple and Protected GSS API Negotiation Mechanism (SPNEGO)
Transmission Control Protocol (TCP)
Transport Layer Security (TLS)
Uniform Resource Locator (URL)

The following terms are specific to this document:

block: A subdivision of a **segment**, when divided into units of equal size, such as 64KB. The last block can be smaller, if the block size is not a multiple of a standard **segment** size.

block hash: A hash of a **content block** within a **segment**.

client: The entity that initiates communication with the **hosted cache**, to offer it **segments** of data.

client-role peer: A **peer** that is looking for **content**, either from the server or from other **peers** or **hosted caches**.

content: A file that an application wants to access. Examples of content include web pages and documents stored on web servers or file servers.

content server: The original source of the **content** that **peers** retrieve from each other.

HoD: The hash of the **content block hashes** of every **block** in the **segment**.

HoHoDk: A hash that represents the content-specific label or public identifier that is used to discover **content** from other **peers** or from the **hosted cache**. This identifier is disclosed freely in broadcast messages. Knowledge of this identifier does not prove authorization to access the actual **content**.

hosted cache: A centralized cache comprised of **blocks** added by **peers**.

peer: A node that both accesses the **content** and serves the **content** it caches for other peers.

Peer Content Caching and Retrieval: Retrieval Protocol (PCCRR): The Peer Content Caching and Retrieval: Retrieval Protocol [MS-PCCRR].

segment: A unit of **content**. **Content** is divided into segments of equal size, except the last segment, which can be smaller, if the **content** size is not a multiple of the standard segment sizes.

segment hash of data: See **HoD**.

segment secret: The **content**-specific hash that is sent to authorized **clients** along with the rest of the **content** information. It is generated by hashing the concatenation of the **HoD** and the server-configured secret.

server-role peer: A **peer** that listens for incoming **block**-range requests from **client-role peers** and responds to the requests.

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

References to Microsoft Open Specification documents do not include a publishing year because links are to the latest version of the documents, which are updated frequently. References to other documents include a publishing year when one is available.

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-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-PCCRC] Microsoft Corporation, "Peer Content Caching and Retrieval: Content Identification".

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

[MS-SPNG] Microsoft Corporation, "Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) Extension".

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

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000, http://www.ietf.org/rfc/rfc2743.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt

[RFC4559] Jaganathan, K., Zhu, L., and Brezak, J., "SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows", RFC 4559, June 2006, http://www.ietf.org/rfc/rfc4559.txt

1.2.2 Informative References

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

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

1.3 Overview

The Peer Content Caching and Retrieval: Hosted Cache Protocol provides a mechanism for **clients** to inform the **hosted cache** about **segment** availability. There are two primary roles:

- Client: The client informs the hosted cache that it has segments it can offer.
- Hosted cache: The hosted cache gets the range of **block hashes** associated with the segment being offered, and then retrieves the blocks within the segment that it actually needs.

1.4 Relationship to Other Protocols

The client's connection to the hosted cache uses **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)** ([RFC2818]) as a transport. The **content** encoding used when the client offers the segment and associated **blocks** to the hosted cache follows the format specified in the **Peer Content Caching and Retrieval: Retrieval Protocol (PCCRR)** [MS-PCCRR].

The hosted cache uses the PCCRR framework as a **client-role peer** to retrieve the blocks from the **peer** that is offering them.

1.5 Prerequisites/Preconditions

The following are prerequisites for using this protocol:

- The protocol client is required to have a set of blocks within a segment that it can offer to the hosted cache. Typically, these blocks are retrieved by a higher layer from the **content server**. The higher layer then provides these blocks to this protocol to offer to the hosted cache.
- The hosted cache is required to be provisioned with a certificate chain and associated private key, and the client with the chain's root certificate, such that both are compatible with HTTPS Server authentication (see [RFC2818]).
- The client is initialized by explicitly provisioning it with the fully qualified **DNS** name and the **TCP** port number of the hosted cache.
- The hosted cache is initialized by starting to listen for incoming **HTTP** requests on the **URL** specified in section 2.1.
- If the hosted cache is configured to require client authentication, both the client and the hosted cache are required to support **SPNEGO**-based HTTP authentication ([RFC4559] and [MS-SPNG]) within the HTTPS transport.
- The client is an actively listening **server-role peer**, as described in the Peer Content Caching and Retrieval: Retrieval Protocol (PCCRR) framework [MS-PCCRR]. The port it is listening on is passed as part of the CONNECTION_INFORMATION field in the various request messages from the client to the hosted cache. This allows the hosted cache to use the PCCRR framework to connect to the client to retrieve data blocks in the segment.

1.6 Applicability Statement

Enterprise branch offices typically connect to headquarters over low-bandwidth/high-latency wide area network (WAN) links. As a result, WAN links are generally congested, and application responsiveness in the branch is poor as well. To increase responsiveness, the hosted cache is placed in the branch. The hosted cache then caches content, and serves that content to peers in the branch that request it.

Data gets added to the hosted cache by clients in the branch. Clients check to see if data is available in the hosted cache; if not, they retrieve data from the content server across the WAN link and subsequently add it to the hosted cache.<1>

1.7 Versioning and Capability Negotiation

There is no version negotiation or capability negotiation behavior.

- Supported Transports: This protocol is implemented on top of HTTPS.
- Protocol Versions: The protocol version is 1.0.
- Security and Authentication Methods: A client authenticates the hosted cache using HTTPS, which provides encryption and data integrity verification for data on the wire. In addition, the hosted cache can authenticate clients using the mechanisms described in [RFC4559], which are based on **GSS**-API [RFC2743].
- Localization: Localization-dependent protocol behavior is specified in sections 2.2 and 3.1.5.

1.8 Vendor-Extensible Fields

There are no vendor-extensible fields.

1.9 Standards Assignments

Parameter	Value	Reference
Port	443	
URL	https://:<port number>/C574AC30-5794-4AEE-B1BB-6651C5315029	[RFC2818]

2 Messages

2.1 Transport

The Peer Content Caching and Retrieval: Hosted Cache Protocol uses a client/server transport built on top of Hypertext Transfer Protocol (HTTP) over **Transport Layer Security (TLS)** (HTTPS) [RFC2818].

The client sends a request message as the payload of an HTTP POST request, and the server sends the response message as payload of the HTTP response.

The URL on which the server MUST listen is https://:<port number>/C574AC30-5794-4AEE-B1BB-6651C5315029. The port number used SHOULD be 443,<2> but a higher-layer action such as an administrator MAY specify a different legal port number. In that case, the higher-layer action MUST provide the client with the correct port number of the hosted cache.

The client MUST be configured with the location, including machine name and port number, of the hosted cache that it will connect to when it has content to offer.

The hosted cache can be configured to require SPNEGO-based HTTP authentication [RFC4559] of the client. If so configured, the hosted cache MUST respond to an HTTP request message lacking an acceptable authorization header with a response indicating a 401 status code. In that case, the transport MUST pass that status code to the protocol layer.

2.2 Message Syntax

This protocol references commonly used data types as defined in [MS-DTYP].

2.2.1 Request Messages

Request messages consist of a message header, connection information, and a message body.

The general request message structure is shown below.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageHeader																															
...																															
ConnectionInformation																															
...																															
MessageBody (variable)																															
...																															

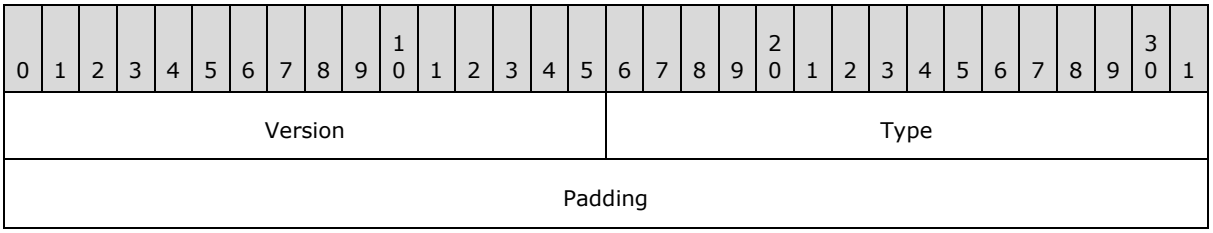
MessageHeader (8 bytes): A MESSAGE_HEADER structure (section 2.2.1.1).

ConnectionInformation (8 bytes): A CONNECTION_INFORMATION structure (section 2.2.1.2).

MessageBody (variable): The message payload, which is specific to the type of message identified in the **MessageHeader** field.

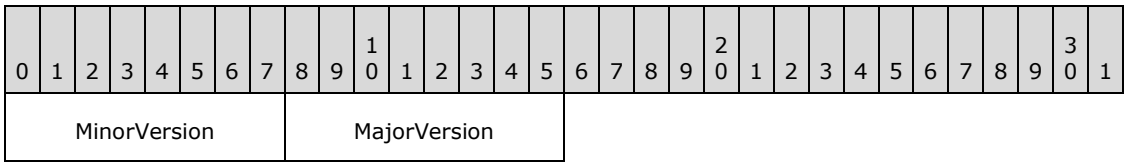
2.2.1.1 MESSAGE_HEADER

Request messages use a common header, which consists of the following fields.



Version (2 bytes): The message **version**, expressed as major and minor values. The version MUST be "1.0".

Note that the order of the subfields is reversed; the **MinorVersion** subfield occurs first.



MinorVersion (1 byte): The minor part of the version, which MUST be 0x00.

MajorVersion (1 byte): The major part of the version, which MUST be 0x01.

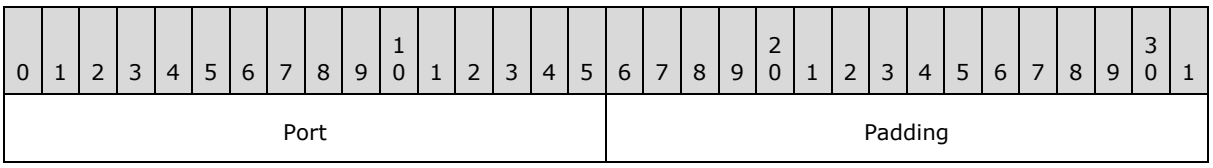
Type (2 bytes): A 16-bit unsigned integer that specifies the message type.

Value	Meaning
0x0001	The message is an INITIAL_OFFER_MESSAGE (section 2.2.1.3).
0x0002	The message is a SEGMENT_INFO_MESSAGE (section 2.2.1.4).

Padding (4 bytes): The value of this field is indeterminate and MUST be ignored on processing.

2.2.1.2 CONNECTION_INFORMATION

Request messages use a common connection information structure, which describes the information needed by the hosted cache to use the Peer Content Caching & Retrieval: Retrieval Protocol [MS-PCCRR] as a client-role peer, to retrieve needed blocks from the client as a server-role peer.



...

Port (2 bytes): A 16-bit unsigned integer that MUST be set by the client to the port on which it is listening as a server-role peer, for use with the retrieval protocol.

Padding (6 bytes): The value of this field is indeterminate and MUST be ignored on processing.

2.2.1.3 INITIAL_OFFER_MESSAGE

An INITIAL_OFFER_MESSAGE is the first message a client sends to the hosted cache. The INITIAL_OFFER_MESSAGE is a request message that notifies the hosted cache of new content available on the client.

An INITIAL_OFFER_MESSAGE consists of the following fields.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
MessageHeader																															
...																															
ConnectionInformation																															
...																															
Hash (variable)																															
...																															

MessageHeader (8 bytes): A MESSAGE_HEADER structure (section 2.2.1.1), with the **Type** field set to 0x0001.

ConnectionInformation (8 bytes): A CONNECTION_INFORMATION structure (section 2.2.1.2).

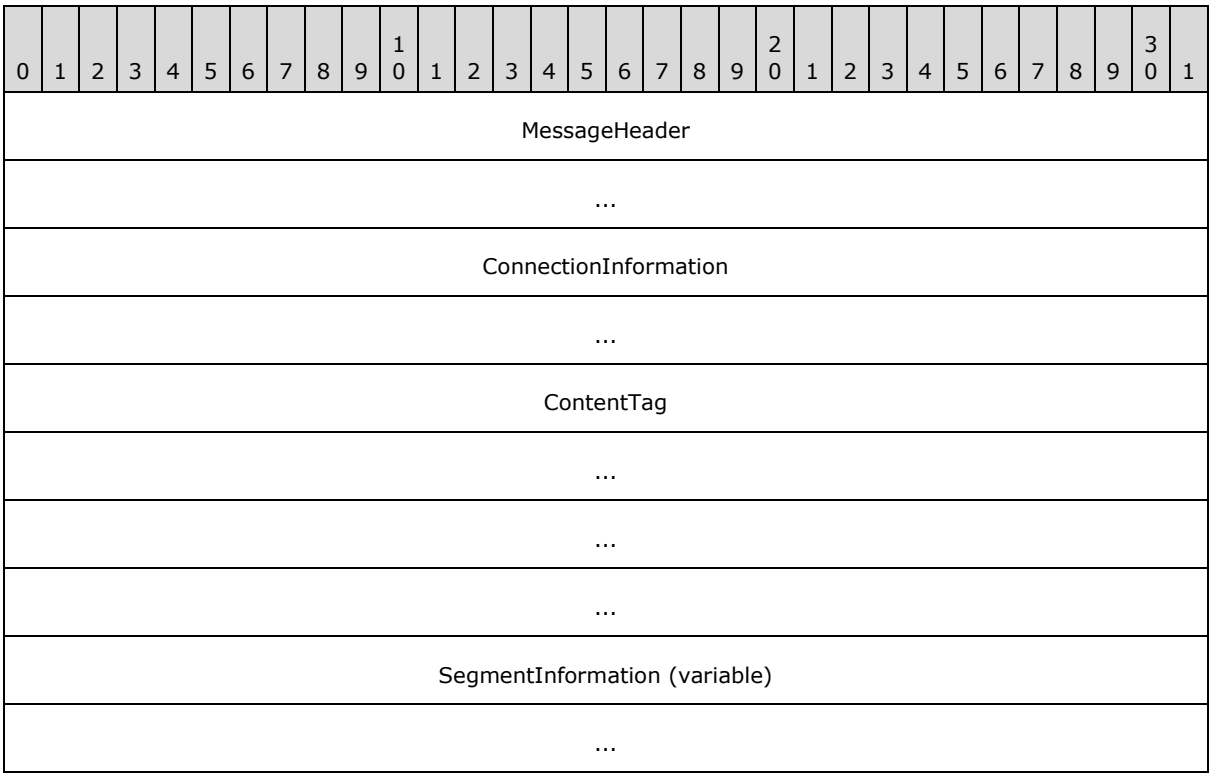
Hash (variable): The **Hash** field is a binary byte array that contains the **HoHoDk** of the segment that was partly or fully retrieved by the client.

The size of this field is calculated as the total message size minus the sum of the field sizes that precede the **Hash** field.

2.2.1.4 SEGMENT_INFO_MESSAGE

A SEGMENT_INFO_MESSAGE is a request message sent by the client to the hosted cache containing the **segment hash of data** (**HoD**) for the previously offered segment, as well as the range of block hashes in the segment. Whether a SEGMENT_INFO_MESSAGE is sent depends on the hosted cache's response to the previous INITIAL_OFFER_MESSAGE containing the same HoHoDk.

A SEGMENT_INFO_MESSAGE consists of the following fields.



MessageHeader (8 bytes): A MESSAGE_HEADER structure (section 2.2.1.1), with the **Type** field set to 0x0002.

ConnectionInformation (8 bytes): A CONNECTION_INFORMATION structure (section 2.2.1.2).

ContentTag (16 bytes): A structure consisting of 16 bytes of opaque data.

This field contains a tag supplied by a higher protocol layer on the client. The tag is added to the information being sent by the client to the hosted cache. The data is then passed to the higher-layer application on the hosted cache.

SegmentInformation (variable): A Content Information data structure ([MS-PCCRC] section 2.3).

This field describes the single segment being offered, with information retrieved from the content server. The **SegmentInformation** field also contains the subfields of the segment's Content Information data structure, SegmentDescription, and SegmentContentBlocks, as specified in [MS-PCCRC] sections 2.3.1.1 and 2.3.1.2, respectively.

- The **Version** and **dwHashAlgo** fields MUST be copied directly from the client's Content Information data structure for the content containing the segment being offered.
- The **dwOffsetInFirstSegment** field MUST be set to the offset in the segment being offered at which the content range begins.
- The **dwReadBytesInLastSegment** field MUST be set to the total number of bytes in the segment being offered.

- The **cSegments** field MUST be set to 1.
- The **segments** field MUST contain the single SegmentDescription ([MS-PCCRC] section 2.3.1.1) in the original Content Information data structure corresponding to the segment being offered.
- The **blocks** field MUST contain a single SegmentContentBlocks ([MS-PCCRC] section 2.3.1.2) corresponding to the segment being offered, copied from the **blocks** field in the original Content Information data structure.

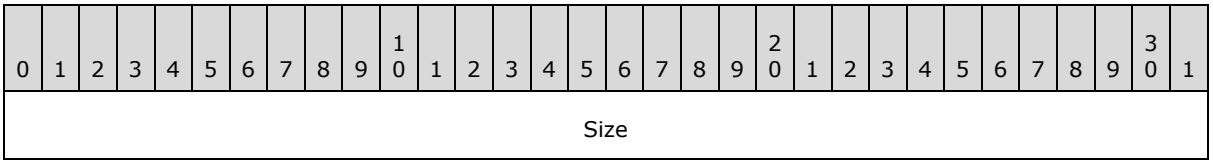
2.2.2 Response Messages

Response messages are sent in response to the following request messages:

- INITIAL_OFFER_MESSAGE, section 2.2.1.3
- SEGMENT_INFO_MESSAGE, section 2.2.1.4

2.2.2.1 Transport Header

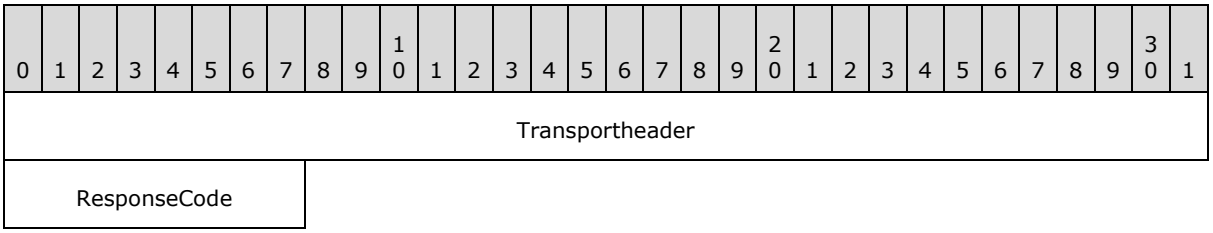
The transport adds the following header in front of any response protocol message.



Size (4 bytes): Total message size in bytes, excluding this field.

2.2.2.2 Response Code

Each response message contains a response code, as specified below.



Transportheader (4 bytes): A transport header (section 2.2.2.1).

ResponseCode (1 byte): A code that indicates the server response to the client request message.

Value	Meaning
OK 0x00	The server has sufficient information to retrieve content from the client.
INTERESTED 0x01	The server needs the range of block hashes from the client before it can retrieve content from the client.

In an INITIAL_OFFER_MESSAGE (section 2.2.1.3), the response code MUST be either **OK** or **INTERESTED**. In a SEGMENT_INFO_MESSAGE (section 2.2.1.4), the response code MUST be **OK**.

3 Protocol Details

There are two roles in the Peer Content Caching and Retrieval: Hosted Cache Protocol, the hosted cache and the client.

3.1 Server Details

The hosted cache is the server entity that is offered a content segment and then determines if it will get more information about the block hashes contained in that segment.

3.1.1 Abstract Data Model

The following state is maintained for the operation of the hosted cache:

- **Content information for the offered segment**: This is comprised of the segment HoHoDk, segment HoD, and a list of block hashes contained within the segment.
- **Block cache**: A cache of data blocks retrieved from clients, together with their corresponding HoHoDks, segment hashes, block hashes, and the **segment secrets**. The data blocks are made available to other client-role peers that attempt to retrieve them using the Peer Content Caching and Retrieval: Retrieval Protocol (PCCRR) framework [MS-PCCRR].

Note The preceding conceptual data can be implemented using a variety of techniques.

3.1.2 Timers

None.

3.1.3 Initialization

The following initialization of the hosted cache MUST be performed:

- The hosted cache MUST be initialized by starting to listen for incoming HTTP requests on the URL specified in section 2.1.
- The hosted cache MUST be provisioned with a certificate chain and associated private key such that it is compatible with HTTPS server authentication [RFC2818].

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 INITIAL_OFFER_MESSAGE Request Received

The hosted cache MUST respond with a correctly formatted response message, as specified in section 2.2.2.

- The hosted cache MUST specify a response code of 0 if the hosted cache already has all the block hashes in the segment.

  If the hosted cache does not have all the offered data blocks associated with the block hashes in the segment, it MUST initiate the Peer Content Caching and Retrieval: Retrieval Protocol (PCCRR)

framework [MS-PCCRR] as a client-role peer to retrieve the missing blocks from the offering client.

The hosted cache, acting as a PCCRR client-role peer, MUST connect to the client's IP address using the port number specified in the **CONNECTION_INFORMATION** field from the INITIAL_OFFER_MESSAGE request, as specified in section 2.2.1.3. The HoHoDk in the INITIAL_OFFER_MESSAGE request MUST be used to retrieve the corresponding segment hash of data (HoD), list of block hashes, and the segment secret from the hosted cache's block cache (section 3.1.1). The segment HoD, list of block hashes, and the segment secret MUST be passed to the PCCRR client-role peer. The retrieved blocks MUST be added to the hosted cache's block cache.

- The hosted cache MUST specify a response code of 1 if its list of block hashes associated with the segment is incomplete.

3.1.5.2 SEGMENT_INFO_MESSAGE Request Received

Regardless of whether an INITIAL_OFFER_MESSAGE has previously been received from this client, the hosted cache MUST respond with a message formatted as specified in section 2.2.2 and MUST perform the following actions:

- Send a response code of 0.
- Initiate the Peer Content Caching and Retrieval: Retrieval Protocol (PCCRR) framework [MS-PCCRR] as a client-role peer to retrieve the missing blocks from the offering client.
- The hosted cache, acting as a PCCRR client-role peer, MUST connect to the client's IP address using the port number specified in the **CONNECTION_INFORMATION** field from the SEGMENT_INFO_MESSAGE request, as specified in section 2.2.1.4. The segment hash of data (HoD), list of block hashes, and the segment secret from the **SegmentInformation** field (section 2.2.1.4) MUST be passed to the PCCRR client-role peer. The retrieved blocks MUST be added to the hosted cache's block cache.
- The **ContentTag** MUST be passed to the higher layer. The **ContentTag** is described in the SEGMENT_INFO_MESSAGE request, section 2.2.1.4.

3.1.5.3 Other Message Received

If anything other than a correctly formatted INITIAL_OFFER_MESSAGE (section 2.2.1.3) or SEGMENT_INFO_MESSAGE (section 2.2.1.4) is received, it MUST be dropped and the protocol sequence aborted.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

The client initiates the use of this protocol once there are new blocks available to offer to the hosted cache.

3.2.1 Abstract Data Model

The following state is maintained for the operation of the client:

- **Content information for the offered segment**: This is comprised of the segment HoHoDk, segment HoD, and a list of block hashes contained within the segment. The segment HoHoDk is used in an INITIAL_OFFER_MESSAGE (section 2.2.1.3). The segment HoD and the list of block hashes are used in a SEGMENT_INFO_MESSAGE (section 2.2.1.4).
- **Outstanding request messages**: A set of pending request messages whose timer has not yet expired. For the INITIAL_OFFER_MESSAGE, the HoHoDk that is used is stored along with the request. This allows the client to send the corresponding segment HoD and block hashes in a subsequent SEGMENT_INFO_MESSAGE.
- **Cache**: A cache of data blocks associated with the segment/blocks being offered. This cache includes a mapping between the block hashes/segment hashes and the actual data blocks themselves. These blocks can later be retrieved by the hosted cache using the Peer Content Caching and Retrieval: Retrieval Protocol (PCCRR) framework [MS-PCCRR].
- **Content tag**: The content tag is provided by the higher layer when it initiates the sending of an INITIAL_OFFER_MESSAGE. It is stored for use in the **ContentTag** field of a subsequent SEGMENT_INFO_MESSAGE in case that message is sent. The value of the content tag is determined by the implementation.<3>

Note The preceding conceptual data can be implemented using a variety of techniques.

3.2.2 Timers

Request Timer: For each request message it sends to the hosted cache, a client sets a request timer that expires after 5 seconds. This timer is distinct for each request sent.

3.2.3 Initialization

The client initialization MUST explicitly include the following information:

- The fully qualified DNS name and the TCP port of the hosted cache.
- The chain's root certificate such that it is compatible with HTTPS server authentication [RFC2818].
- The client content information associated with the segment, as described in the section 3.2.1. This content is provided by a higher protocol layer.

3.2.4 Higher-Layer Triggered Events

New blocks available:

When the higher layer has new blocks in a segment to offer the hosted cache, it passes them to this protocol, along with the segment's associated content information and the content tag. The client SHOULD construct an INITIAL_OFFER_MESSAGE request message (section 2.2.1.3) that contains the segment HoHoDk, send it to the hosted cache, store it along with the content tag in its set of outstanding request messages, and start the request timer.

The higher layer SHOULD initiate the use of this protocol only when a sufficient number of new blocks have been received from the content server. Doing otherwise, such as initiating the protocol for every new block that becomes available, could lead to poor network performance.<4>

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 INITIAL_OFFER_MESSAGE Response Received

If a message response matches one of its set of outstanding requests, the client MUST delete it from the set of outstanding requests and cancel the request timer for this request. If the response is "INTERESTED", the client MUST respond with a SEGMENT_INFO_MESSAGE request (section 2.2.1.4) for the associated HoHoDk, which MUST be stored in its set of outstanding request messages. A request timer must also be set for this message.

If there are no outstanding requests that match with the message response, the client MUST discard the message.

3.2.5.2 SEGMENT_INFO_MESSAGE Response Received

The client MUST cancel the request timer for the corresponding request and remove it from the client's set of outstanding request messages.

3.2.5.3 HTTP Status Code 401 Response Received

The client MUST resend the request, indicating to the transport that SPNEGO-based HTTP authentication should be performed (section 2.1). The request timer for the request must also be reset to its initial expiration time.

3.2.5.4 Other Message Received

If anything other than a correctly formatted INITIAL_OFFER_MESSAGE request (section 2.2.1.3) or SEGMENT_INFO_MESSAGE request (section 2.2.1.4) response is received on the port the client is currently using for this protocol, it MUST be dropped and the protocol sequence aborted.

3.2.6 Timer Events

Request timer expires: The related outstanding message request MUST be removed.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 Hosted Cache with No Block Hashes

This section presents an example of a hosted cache that has none of the block hashes associated with the segment that is offered.

In this sequence, on availability of new blocks for a segment, the client uses the protocol to offer the associated segment to the hosted cache. The hosted cache determines that it has no block hashes, and therefore requests that the client send it complete information on the segment, so that the hosted cache can then use the Peer Content Caching & Retrieval: Retrieval Protocol [MS-PCCRR] to retrieve the blocks desired.

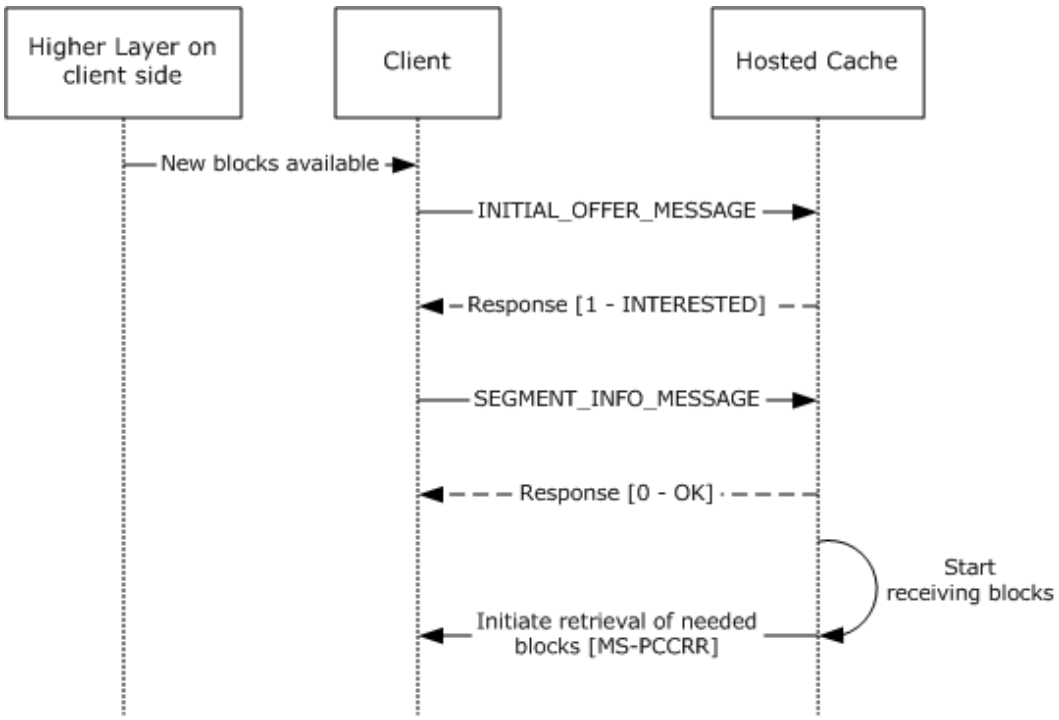


Figure 1: Hosted cache with no block hashes

4.2 Hosted Cache with Block Hashes and No Data Blocks

This section presents an example of a hosted cache that has the block hashes associated with the segment but no data blocks.

In this sequence, on availability of new blocks for a segment, the client uses the protocol to offer the associated segment to the hosted cache. The hosted cache determines that it has the block hashes for the segment, but does not have any of the data blocks, and thus responds with a code of zero. At the same time, the hosted cache uses the Peer Content Caching and Retrieval: Retrieval Protocol [MS-PCCRR] to retrieve all blocks of the segment that are available from the offering client.

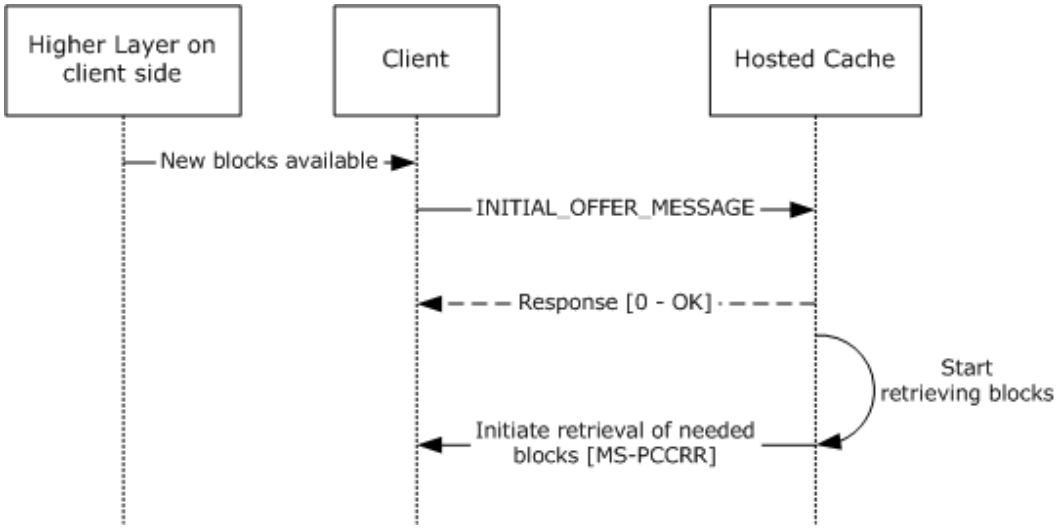


Figure 2: Hosted cache with block hashes and no data blocks

4.3 Hosted Cache with Block Hashes and Data Blocks

This section presents an example of a hosted cache that has all the block hashes associated with the segment and all the data blocks.

In this sequence, on availability of new blocks for a segment, the client uses the protocol to offer the associated segment to the hosted cache. The hosted cache determines that it already has all blocks for the segment and responds with a code of 0.

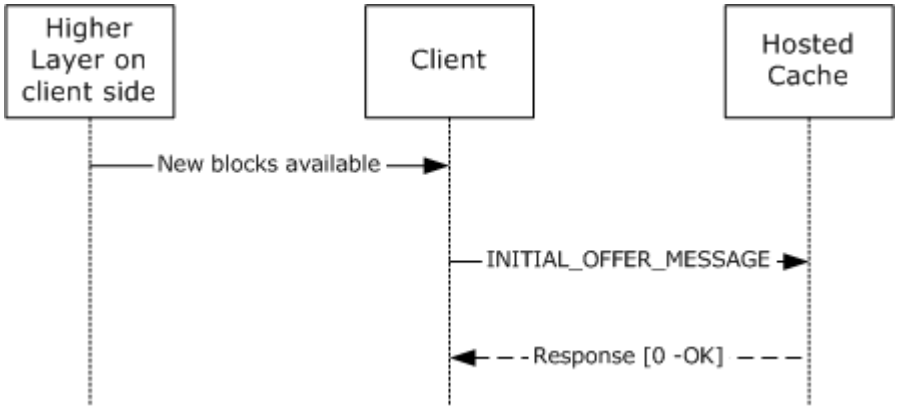


Figure 3: Hosted cache with block hashes and data blocks

5 Security

5.1 Security Considerations for Implementers

Peer Content Caching and Retrieval: Hosted Cache Protocol messages are secured using HTTPS.

An HTTPS connection is established by the client with the hosted cache. In addition, the hosted cache can choose to authenticate clients.<5> This is done using the HTTP authentication mechanism described in [RFC4559].

5.2 Index of Security Parameters

Security Parameter	Section
HTTPS	2.1

6 Appendix A: 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 with the installation of the Background Intelligent Transfer Service (BITS) in the Windows Management Framework. For more information, see [MSDN-BITS]. Support for the server-side elements of this protocol is not available for Windows Vista or Windows Server 2008.

<2> Section 2.1: In a Windows implementation, the hosted cache listens on port 443 by default.

<3> Section 3.2.1: In the Windows implementation, the values of the content tag can be the following:

- The ASCII string "WinINet".
- The ASCII string "WebIO".
- The ASCII string "BITS-4.0".
- The binary byte array {0x35, 0xDB, 0x04, 0x5D, 0x14, 0x23, 0x45, 0x53, 0xA0, 0x51, 0x0D, 0xC2, 0xE1, 0x5E, 0x6C, 0x4C}.

<4> Section 3.2.4: Windows invokes this protocol after 20% of new blocks of a segment have been received from the content server.

<5> Section 5.1: On Windows, by default, the hosted cache authenticates all clients; however, a higher layer or administrator action can configure it to stop authenticating clients.

7 Change Tracking

This section identifies changes that were made to the [MS-PCHC] protocol document between the May 2011 and June 2011 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact protocol@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
1.2 References	Added explanatory statement regarding the removal of the publishing year from Microsoft Open Specification document references.	N	Content updated.

8 Index