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

Revision Summary

Date	Revision History	Revision Class	Comments
12/05/2008	0.1	Major	Initial Availability
01/16/2009	0.2	Minor	Updated the technical content.
02/27/2009	0.2.1	Editorial	Revised and edited the technical content.
04/10/2009	1.0	Major	Updated and revised the technical content.
05/22/2009	1.1	Minor	Updated the technical content.
07/02/2009	1.1.1	Editorial	Revised and edited the technical content.
08/14/2009	1.2	Minor	Updated the technical content.
09/25/2009	1.3	Minor	Updated the technical content.
11/06/2009	1.4	Minor	Updated the technical content.
12/18/2009	1.5	Minor	Updated the technical content.
01/29/2010	1.6	Minor	Updated the technical content.
03/12/2010	1.6.1	Editorial	Revised and edited the technical content.
04/23/2010	1.6.2	Editorial	Revised and edited the technical content.
06/04/2010	1.6.3	Editorial	Revised and edited the technical content.
07/16/2010	1.6.3	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	1.6.3	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	1.6.4	Editorial	Changed language and formatting in the technical content.
11/19/2010	1.6.4	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	1.6.4	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	1.6.4	No change	No changes to the meaning, language, or formatting of the technical content.
03/25/2011	1.6.4	No change	No changes to the meaning, language, or formatting of the technical content.
05/06/2011	1.6.4	No change	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
06/17/2011	1.7	Minor	Clarified the meaning of the technical content.
09/23/2011	1.7	No change	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	2.0	Major	Significantly changed the technical content.
03/30/2012	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/12/2012	3.0	Major	Significantly changed the technical content.
10/25/2012	4.0	Major	Significantly changed the technical content.
01/31/2013	4.0	No change	No changes to the meaning, language, or formatting of the technical content.
08/08/2013	5.0	Major	Significantly changed the technical content.
11/14/2013	5.0	No change	No changes to the meaning, language, or formatting of the technical content.
02/13/2014	6.0	Major	Significantly changed the technical content.
05/15/2014	6.0	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction 6
1.1 Glossary 6
1.2 References 7
1.2.1 Normative References 7
1.2.2 Informative References 7
1.3 Overview 8
1.4 Relationship to Other Protocols 8
1.5 Prerequisites/Preconditions 8
1.6 Applicability Statement 8
1.7 Versioning and Capability Negotiation 9
1.8 Vendor-Extensible Fields 9
1.9 Standards Assignments 9

2 Messages 10
2.1 Transport 10
2.2 Message Syntax 10
2.2.1 Request Messages 10
2.2.1.1 MESSAGE_HEADER 11
2.2.1.2 CONNECTION_INFORMATION 11
2.2.1.3 INITIAL_OFFER_MESSAGE 12
2.2.1.4 SEGMENT_INFO_MESSAGE 13
2.2.1.5 BATCHED_OFFER_MESSAGE 14
2.2.2 Response Messages 15
2.2.2.1 Transport Header 16
2.2.2.2 Response Code 16

3 Protocol Details 17
3.1 Server Details 17
3.1.1 Abstract Data Model 17
3.1.2 Timers 17
3.1.3 Initialization 17
3.1.4 Higher-Layer Triggered Events 17
3.1.5 Message Processing Events and Sequencing Rules 17
3.1.5.1 INITIAL_OFFER_MESSAGE Request Received 17
3.1.5.2 SEGMENT_INFO_MESSAGE Request Received 18
3.1.5.3 BATCHED_OFFER_MESSAGE Request Received 18
3.1.5.4 Other Message Received 19
3.1.6 Timer Events 19
3.1.7 Other Local Events 19
3.2 Client Details 19
3.2.1 Abstract Data Model 19
3.2.2 Timers 19
3.2.3 Initialization 20
3.2.4 Higher-Layer Triggered Events 20
3.2.5 Message Processing Events and Sequencing Rules 20
3.2.5.1 INITIAL_OFFER_MESSAGE Response Received 20
3.2.5.2 SEGMENT_INFO_MESSAGE Response Received 20
3.2.5.3 HTTP Status Code 401 Response Received 21
3.2.5.4 BATCHED_OFFER_MESSAGE Response Received 21
3.2.5.5 Other Message Received 21

3.2.6 Timer Events 21
3.2.7 Other Local Events 21

4 Protocol Examples 22
4.1 Hosted Cache with No Block Hashes 22
4.2 Hosted Cache with Block Hashes and No Data Blocks 22
4.3 Hosted Cache with Block Hashes and Data Blocks 23
4.4 Hosted Cache with No Data Blocks 23
4.5 Hosted Cache with Data Blocks 24

5 Security 25
5.1 Security Considerations for Implementers 25
5.2 Index of Security Parameters 25

6 Appendix A: Product Behavior 26

7 Change Tracking 28

8 Index 29

1 Introduction

The Peer Content Caching and Retrieval: Hosted Cache Protocol is used by clients to offer metadata to a hosted cache server.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

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**.

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 to be accessed by an application. 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 subdivision of **content**.

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 Specifications documentation 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.

[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".

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.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

[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

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/bb968799(VS.85).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 a hosted cache uses the **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)** ([RFC2818]) or the **Hypertext Transfer Protocol (HTTP)** ([RFC2616]) 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.
- If HTTPS ([RFC2818]) is used as a transport, the hosted cache is required to be provisioned with a certificate and associated private key, and the client with the 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.

1.7 Versioning and Capability Negotiation

There is no version negotiation or capability negotiation behavior. The protocol versions use different transports; the version is implied by the transport used.

- Protocol Versions: The protocol versions are 1.0 and 2.0.<1>
- Supported Transports: Version 1.0 is implemented on top of HTTPS. Version 2.0 is implemented on top of HTTP.
- Security and Authentication Methods: In version 1.0, 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]. In version 2.0, authentication is not employed.
- 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, 80	
URL	https://:443/C574AC30-5794-4AEE-B1BB-6651C5315029 http://:80/0131501b-d67f-491b-9a40-c4bf27bcb4d4	[RFC2818]

2 Messages

2.1 Transport

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

For version 1.0 of the protocol, the URL on which the server MUST listen is https://:<port number>/C574AC30-5794-4AEE-B1BB-6651C5315029 and the port number used SHOULD<2> be 443. For version 2.0, the URL on which the server MUST listen is http://:<port number>/0131501b-d67f-491b-9a40-c4bf27bcb4d4 and the port number used SHOULD be 80. However, a higher-layer action such as that taken by an administrator can 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.

Note The HTTPS URL does not support version 2.0 protocol messages, and the HTTP URL does not support version 1.0 protocol messages.

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.

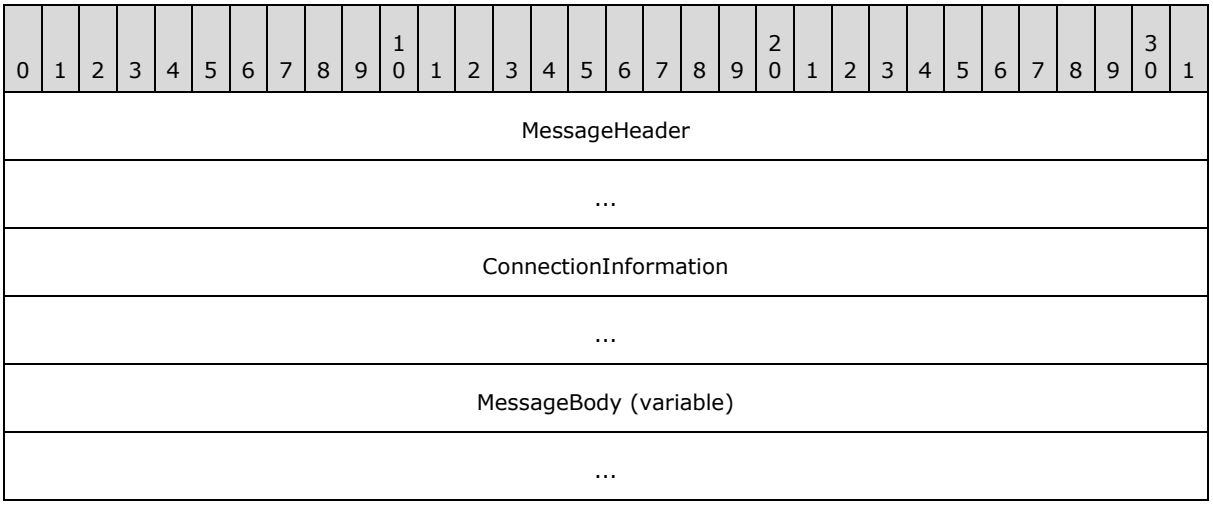
2.2 Message Syntax

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.



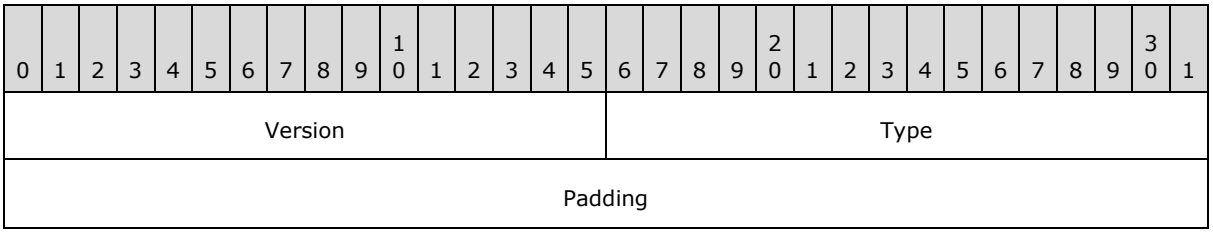
MessageHeader (8 bytes): A MESSAGE_HEADER structure (section 2.2.1.1).

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.

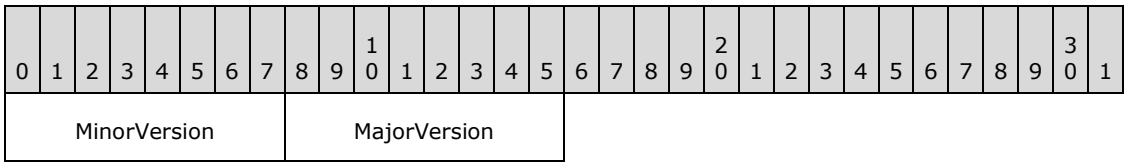
2.2.1.1 MESSAGE_HEADER

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" or "2.0".<3>

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 or 0x02.<4>

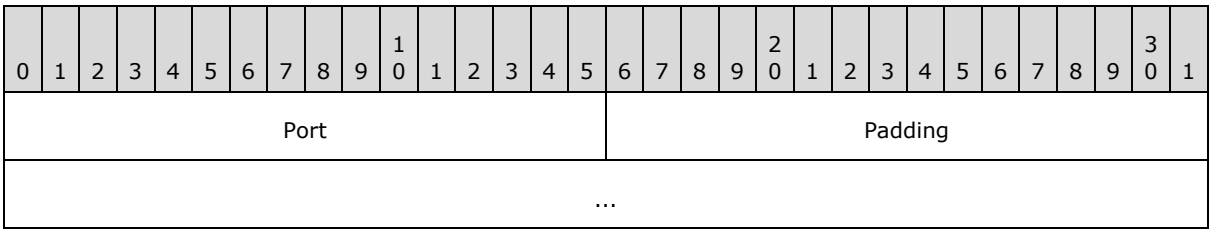
Type (2 bytes): A 16-bit unsigned integer that specifies the message type.

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).
0x0003	The message is a BATCHED_OFFER_MESSAGE (section 2.2.1.5).

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.

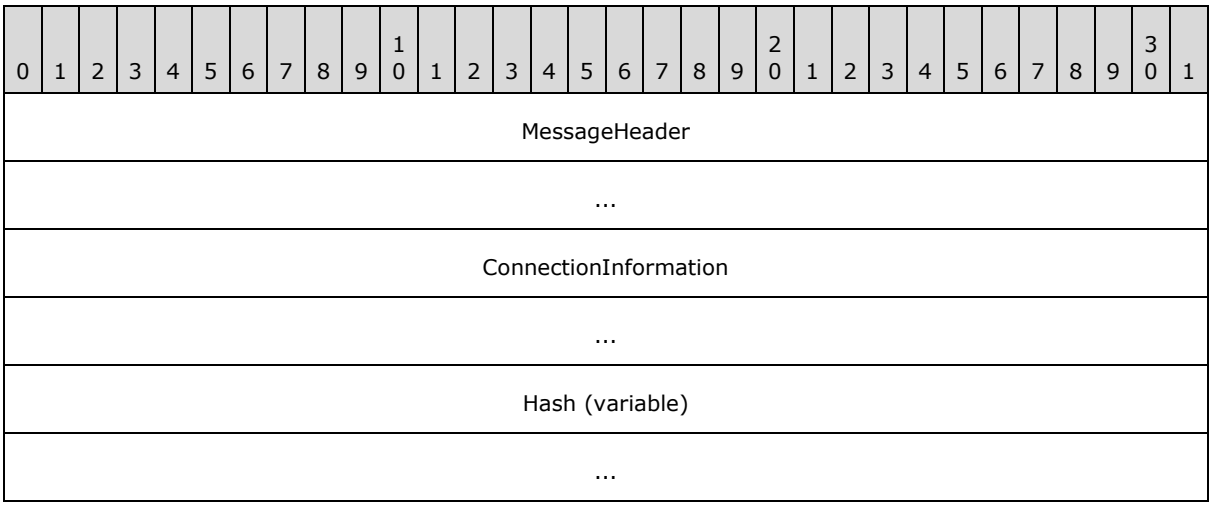
Padding (6 bytes): The value of this field is indeterminate and MUST be ignored on processing.

2.2.1.3 INITIAL_OFFER_MESSAGE

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

Note This message is only available for protocol version 1.0.

An INITIAL_OFFER_MESSAGE consists of the following fields.



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.

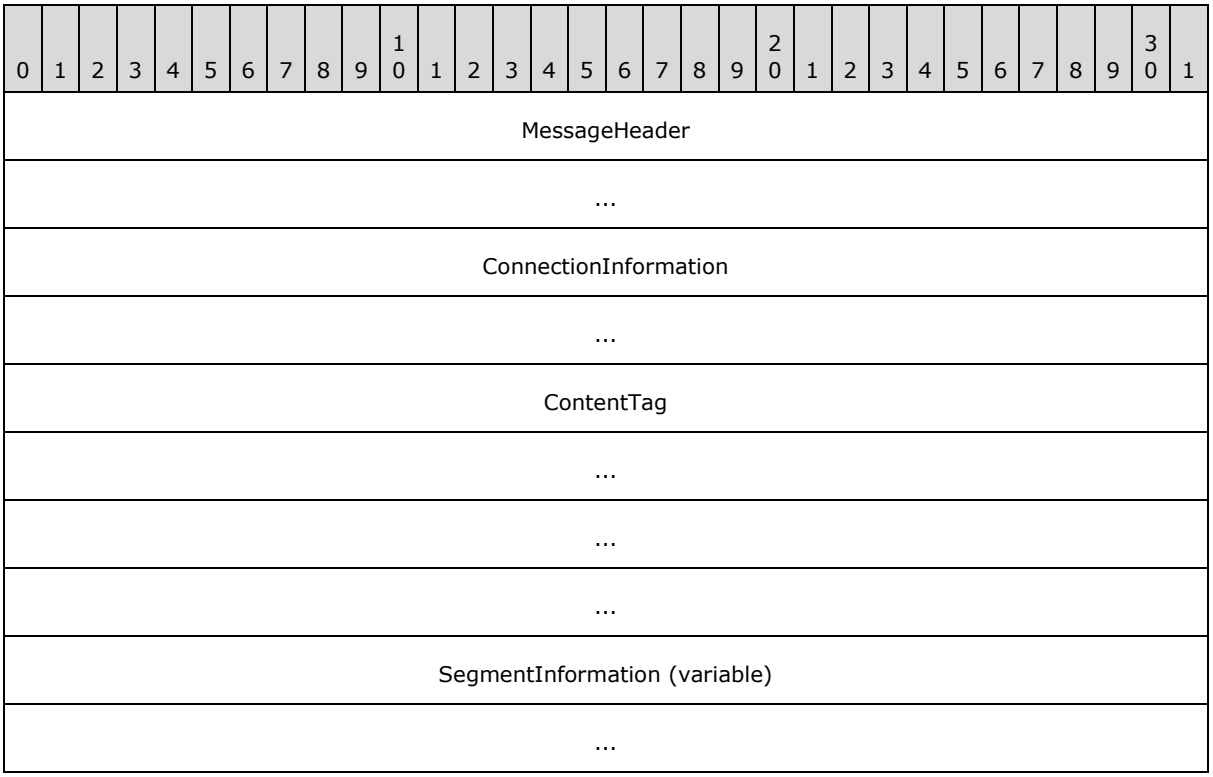
The size of this field is calculated as the total message size minus the sum of the field sizes that precede the **Hash** field.

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.

Note This message is only available for protocol version 1.0.

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-layer protocol 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

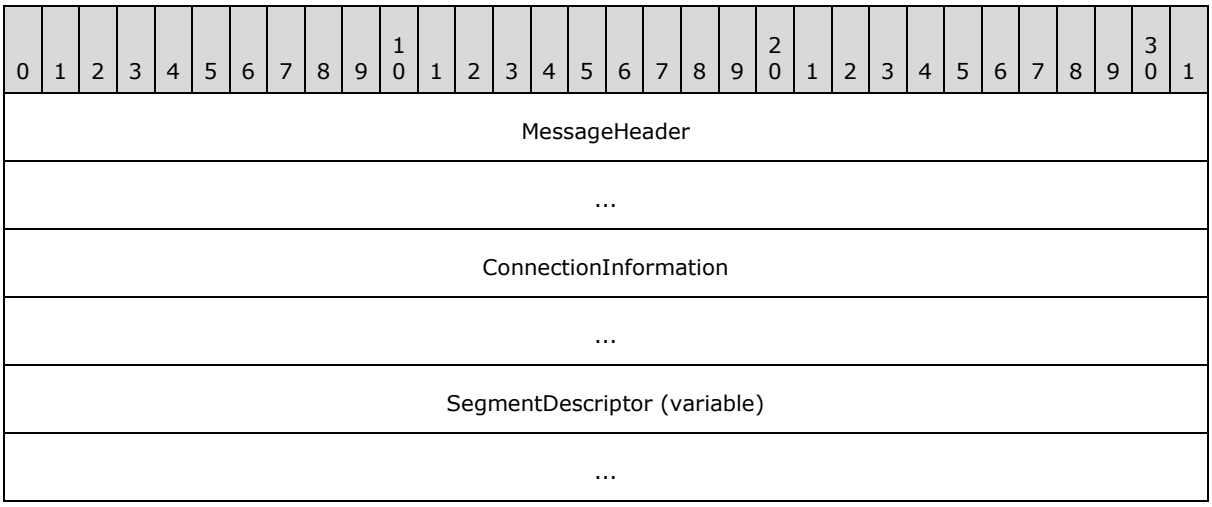
Content Information data structure, SegmentDescription, and SegmentContentBlocks, as specified in [MS-PCCRC] sections 2.3.1.1 and 2.3.1.2, respectively.

- 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.

2.2.1.5 BATCHED_OFFER_MESSAGE

The BATCHED_OFFER_MESSAGE is a request message that notifies the hosted cache of new content that is available on the client.

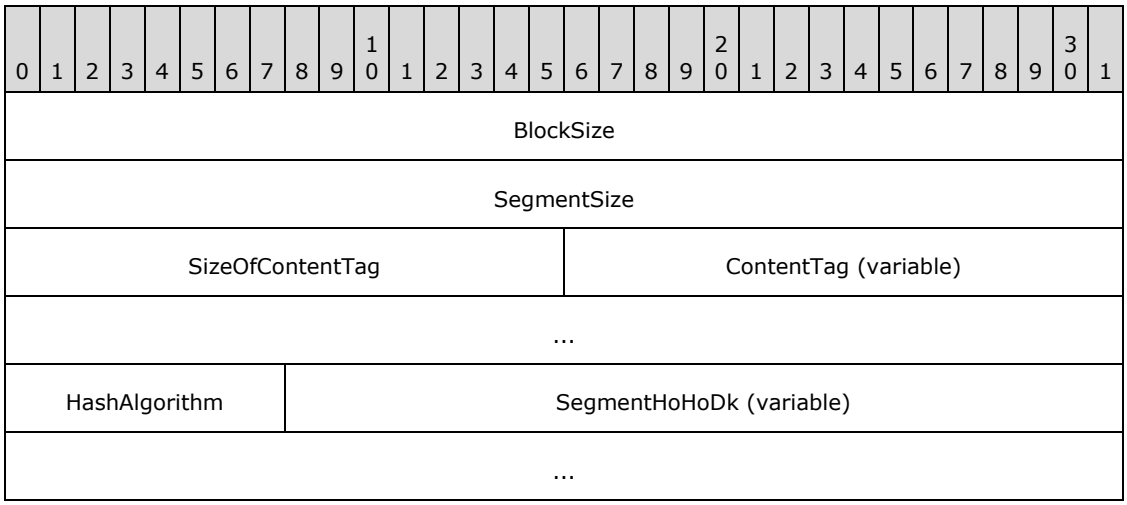
Note This message is only available for protocol version 2.0.



MessageHeader (8 bytes): A MESSAGE_HEADER structure (section 2.2.1.1), with the **Type** field set to 0x0003.

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

SegmentDescriptor (variable): The BATCHED_OFFER_MESSAGE contains a sequence of these segment descriptors. The BATCHED_OFFER_MESSAGE MUST NOT contain more than 128 **SegmentDescriptors**.



BlockSize (4 bytes): Size of each block in the segment.

SegmentSize (4 bytes): Size of data in the segment.

SizeOfContentTag (2 bytes): The size of the data contained in the **ContentTag** field. This field MUST be set to 16 bytes.

ContentTag (variable): Specifies the content tag. The size of this field is indicated by the value of the **SizeOfContentTag** field.

HashAlgorithm (1 byte): The hashing algorithm ID. MUST be one of the following values:

Value	Meaning
0x01	SHA-256
0x04	Truncated SHA-512 (first 256 bits of the SHA-512 hash)

SegmentHoHoDk (variable): Segment identifier; the size of the HoHoDk is indicated by the hashing algorithm ID as shown in the following table.

HashAlgorithm value	Length of SegmentHoHoDk
0x01 (SHA-256)	32 bytes
0x04 (Truncated SHA-512)	32 bytes

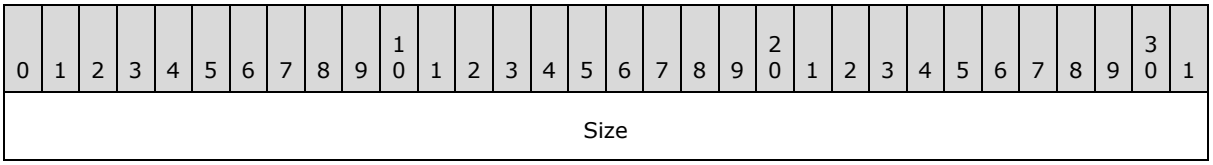
2.2.2 Response Messages

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
- BATCHED_OFFER_MESSAGE, section 2.2.1.5

2.2.2.1 Transport Header

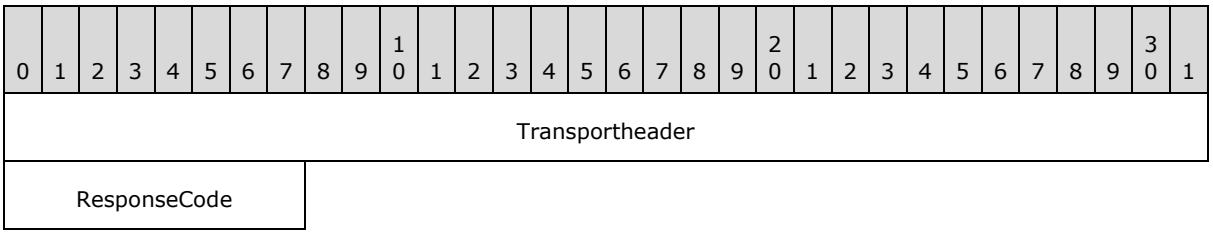
The transport adds the following header in front of any response protocol message.



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 BATCHED_OFFER_MESSAGE (section 2.2.1.5), the response code MUST be **OK**. 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.<5>

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 or URLs specified in section 2.1.
- If HTTPS is used as a transport, the hosted cache MUST be provisioned with a certificate 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 if the request is sent via a registered URL, as specified in section 2.2.2.

- The hosted cache MUST specify a response code of zero 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 0x00.
- 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 BATCHED_OFFER_MESSAGE Request Received

The hosted cache MUST respond with a correctly formatted response message if the request is sent via an HTTP URL, as specified in section 2.2.2.

The hosted cache MUST specify a response code of 0x00.

If the hosted cache does not have all the offered data blocks associated with the segments, 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 by using the port number specified in the **ConnectionInformation** field from the BATCHED_OFFER_MESSAGE request, as specified in section 2.2.1.5. The HoHoDks in the BATCHED_OFFER_MESSAGE request MUST be used to retrieve the corresponding data blocks. The retrieved blocks MUST be added to the hosted cache's block cache.

The **Content tag** (see section 3.2.1) originating from the higher-layer component on the client MUST be passed to the higher layer.

3.1.5.4 Other Message Received

If anything other than a correctly formatted INITIAL_OFFER_MESSAGE (section 2.2.1.3), SEGMENT_INFO_MESSAGE (section 2.2.1.4) or BATCHED_OFFER_MESSAGE (section 2.2.1.5) 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.<6>

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 composed 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) or BATCHED_OFFER_MESSAGE (section 2.2.1.5). 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. For the BATCHED_OFFER_MESSAGE, the HoHoDk list 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 or a BATCHED OFFER_MESSAGE. It is sent in the BATCHED OFFER_MESSAGE and 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. Each higher-layer component SHOULD select a unique content tag and use it for all content offered.<7>

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

3.2.2 Timers

Request Timer: The client uses a request timer to track the expiration of a request message. When a new request message is sent to the hosted cache, the timer is started, or restarted if it was stopped. Once the timer is started, it increments a tick counter every 5 seconds. For each request message sent to the hosted cache, the expiry of that message is calculated as:

Request message expiry = Current tick counter value + 2

Each time the request timer increments the tick counter, the timer checks if the value of the current tick counter has exceeded the expiry of the request message, as shown in the calculation. If the request message is found to have expired, the client drops the expired message and does not process any response messages for the expired message that arrive after the message has been dropped. When there are no more pending outbound requests, the request timer is stopped.

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-layer protocol.

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) or BATCHED OFFER_MESSAGE request message (section 2.2.1.5) that contains the segment HoHoDks, 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.<8>

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 to perform SPNEGO-based HTTP authentication (section 2.1). The request timer for the request MUST also be reset to its initial expiration time.

3.2.5.4 BATCHED_OFFER_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.

3.2.5.5 Other Message Received

If any message other than a correctly formatted INITIAL_OFFER_MESSAGE (section 2.2.1.3) request, SEGMENT_INFO_MESSAGE (section 2.2.1.4) request, or BATCHED_OFFER_MESSAGE (section 2.2.1.5) request response is received on the port which the client is currently using for this protocol, the message 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.

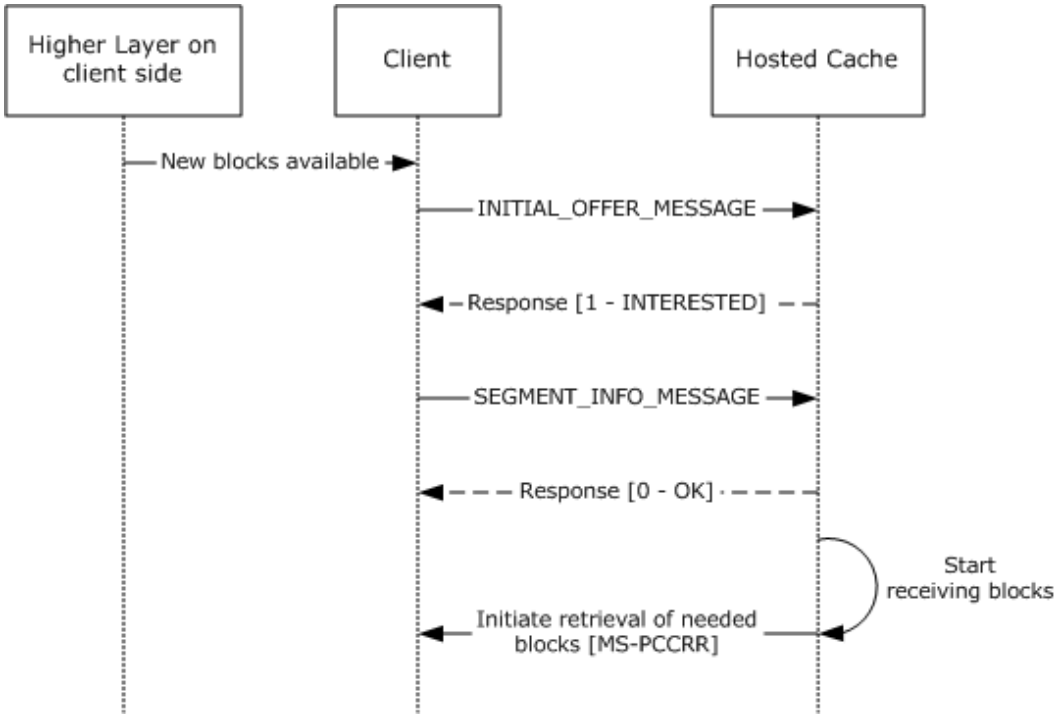


Figure 1: Hosted cache with no block hashes

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.

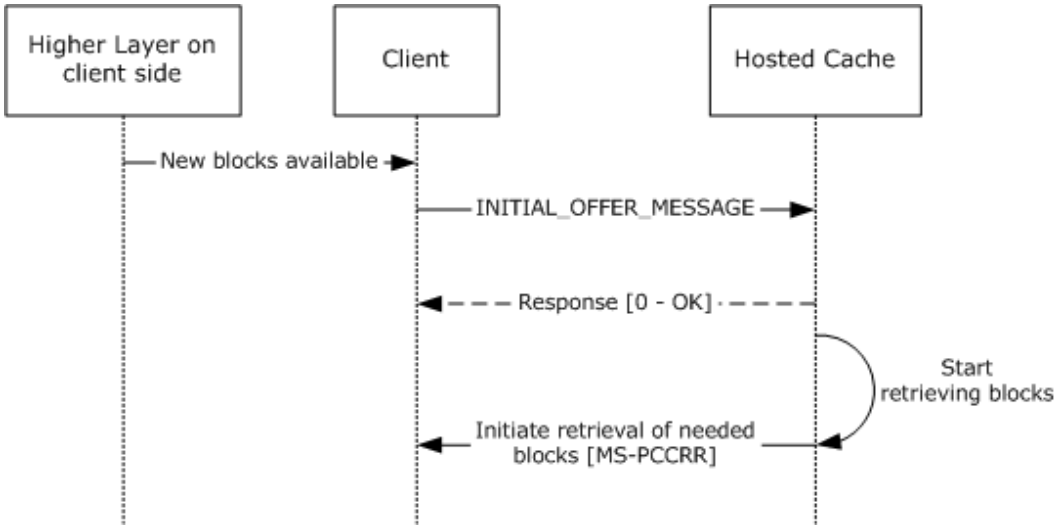


Figure 2: Hosted cache with block hashes and no data blocks

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 zero.

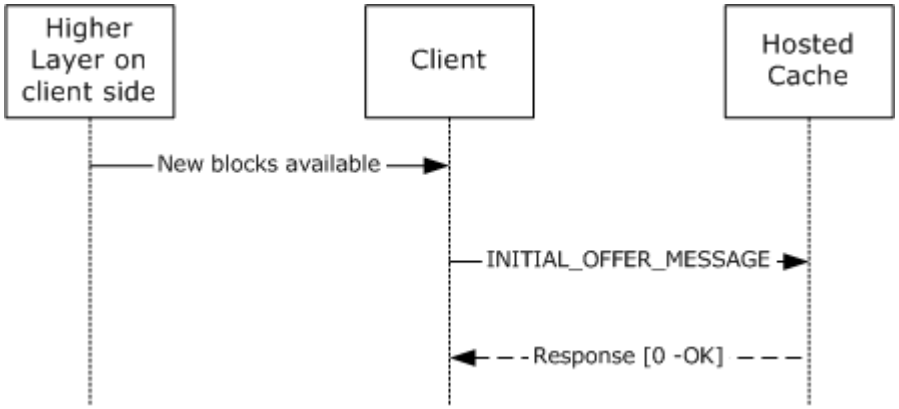


Figure 3: Hosted cache with block hashes and data blocks

4.4 Hosted Cache with No Data Blocks

This section presents an example of a version 2.0 hosted cache that has no data blocks and a version 2.0 client.

In this sequence, upon the availability of new segments, the client uses this protocol to offer the associated segments to the hosted cache. The hosted cache determines that it does not have any of the blocks and thus responds with a code of 0. At the same time, the hosted cache uses the Peer

Content Caching and Retrieval: Retrieval Protocol (PCCRR) [MS-PCCRR] to retrieve blocks of the segments that are available from the offering client.

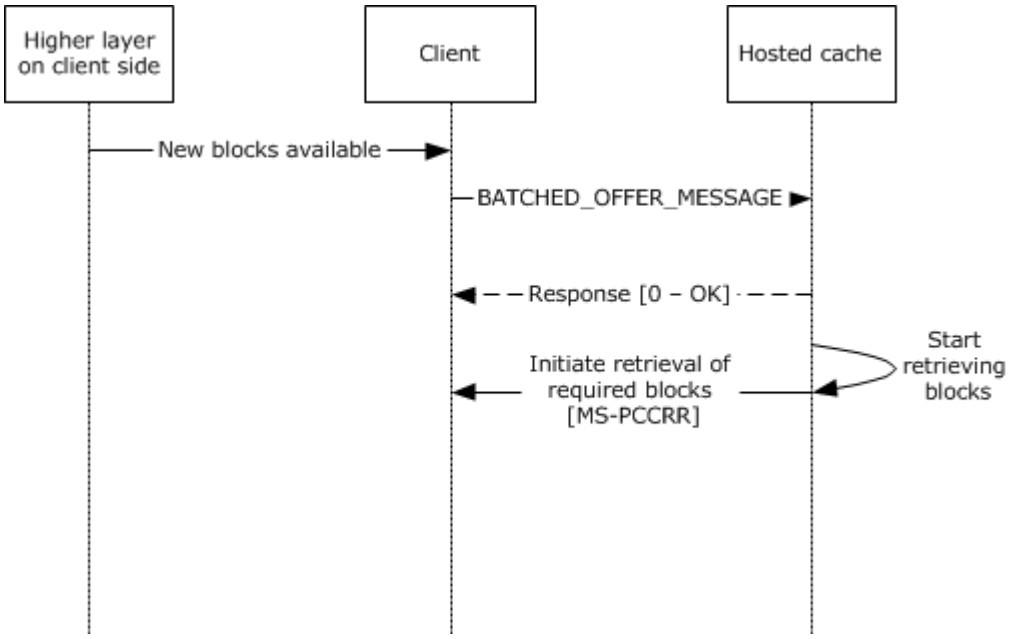


Figure 4: Hosted cache with block hashes and no data blocks

4.5 Hosted Cache with Data Blocks

This section presents an example of a version 2.0 hosted cache that has all the data blocks associated with all the segments in the list and a version 2.0 client.

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

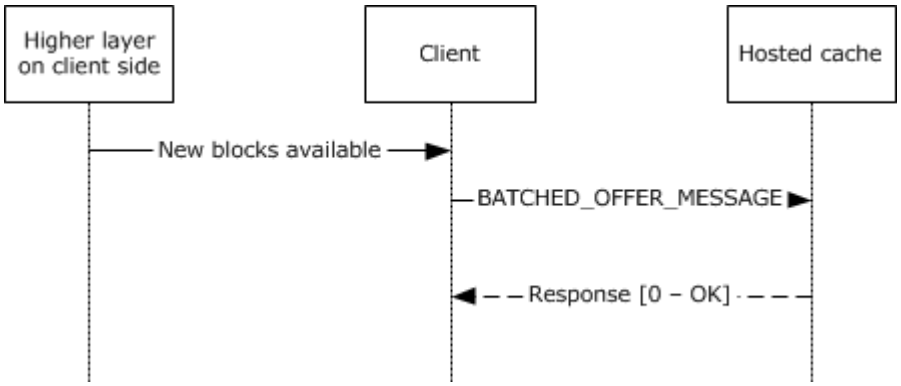


Figure 5: Hosted cache with block hashes and data blocks

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, where the hosted cache can choose to authenticate clients and a higher layer or administrator action can configure it to stop authenticating clients. HTTP authentication is the mechanism used to complete these actions as 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
- Windows 8 operating system
- Windows Server 2012 operating system
- Windows 8.1 operating system
- Windows Server 2012 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 1.7: Windows 7 supports version 1.0; Windows 8 and Windows 8.1 support version 2.0.

<2> Section 2.1: In a Windows Server 2008 R2 implementation, the hosted cache listens on port 443 by default. In a Windows Server 2012 and Windows Server 2012 R2 implementation, the hosted cache listens on both port 80 and port 443 by default.

<3> Section 2.2.1.1: Windows 7 supports version "1.0". Windows 8 and Windows 8.1 support version "2.0".

<4> Section 2.2.1.1: In Windows 7, the value of **MajorVersion** is 0x01. In Windows 8 and Windows 8.1, the value of **MajorVersion** is 0x02.

<5> Section 3.1: For Windows Vista and Windows Server 2008, support for the server-side elements of this protocol is not available.

<6> Section 3.2: 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].

<7> Section 3.2.1: In the Windows implementation, the values of the content tag can be the following:

- The ASCII string "WinINet" is used by clients of the WinInet APIs.
- The ASCII string "WebIO" is used by clients of the WebIO APIs.

- The ASCII string "BITS-4.0" is used by the Background Intelligent Transfer Service.
- The binary byte array {0x35, 0xDB, 0x04, 0x5D, 0x14, 0x23, 0x45, 0x53, 0xA0, 0x51, 0x0D, 0xC2, 0xE1, 0x5E, 0x6C, 0x4C} is used by the SMB stack.

The higher-layer component on cache servers hosted by Windows tracks performance statistics for each of these values and categorizes all other values as "Other".

<8> Section 3.2.4: Windows invokes this protocol after 20% of new blocks of a segment have been received from the content server.

7 Change Tracking

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

8 Index

A

Abstract data model
  client 19
  server 17
Applicability 8

B

BATCHED_OFFER_MESSAGE packet 14

C

Cache - hosted
  with block hashes and data blocks 23
  with block hashes and no data blocks 22
  with data blocks 24
  with no block hashes 22
  with no data blocks 23
Capability negotiation 9
Change tracking 28
Client
  abstract data model 19
  higher-layer triggered events 20
  initialization 20
  local events 21
  message processing
    HTTP status code 401 response received 21
    INITIAL_OFFER_MESSAGE response received 20
    other message received 21
    SEGMENT_INFO_MESSAGE response received 20
  overview 19
  sequencing rules
    HTTP status code 401 response received 21
    INITIAL_OFFER_MESSAGE response received 20
    other message received 21
    SEGMENT_INFO_MESSAGE response received 20
  timer events 21
  timers 19
CONNECTION_INFORMATION packet 11

D

Data model - abstract
  client 19
  server 17

E

Examples
  hosted cache with block hashes and data blocks 23
  hosted cache with block hashes and no data blocks 22
  hosted cache with data blocks 24
  hosted cache with no block hashes 22
  hosted cache with no data blocks 23

F

Fields - vendor-extensible 9

G

Glossary 6

H

Higher-layer triggered events
  client 20
  server 17
Hosted cache
  with block hashes and data blocks 23
  with block hashes and no data blocks 22
  with data blocks 24
  with no block hashes 22
  with no data blocks 23
HTTP status code 401 response received 21

I

Implementer - security considerations 25
Index of security parameters 25
Informative references 7
INITIAL_OFFER_MESSAGE packet 12
Initialization
  client 20
  server 17
Introduction 6

L

Local events
  client 21
  server 19

M

Message processing
  client
    HTTP status code 401 response received 21
    INITIAL_OFFER_MESSAGE response received 20
    other message received 21
    SEGMENT_INFO_MESSAGE response received 20
  server
    BATCHED_OFFER_MESSAGE request received 18
    INITIAL_OFFER_MESSAGE request received 17
    other message received 19
    SEGMENT_INFO_MESSAGE request received 18
MESSAGE_HEADER packet 11
Messages

syntax 10
transport 10

N

Normative references 7

O

Overview (synopsis) 8

P

Parameter index - security 25
Preconditions 8
Prerequisites 8
Product behavior 26

R

References
informative 7
normative 7
Relationship to other protocols 8
RequestMessage packet 10
Response messages 15
ResponseMessage packet 16

S

Security
implementer considerations 25
parameter index 25
SEGMENT_INFO_MESSAGE packet 13
Sequencing rules
client
HTTP status code 401 response received 21
INITIAL_OFFER_MESSAGE response received 20
other message received 21
SEGMENT_INFO_MESSAGE response received 20
server
BATCHED_OFFER_MESSAGE request received 18
INITIAL_OFFER_MESSAGE request received 17
other message received 19
SEGMENT_INFO_MESSAGE request received 18
Server
abstract data model 17
higher-layer triggered events 17
initialization 17
local events 19
message processing
BATCHED_OFFER_MESSAGE request received 18
INITIAL_OFFER_MESSAGE request received 17
other message received 19
SEGMENT_INFO_MESSAGE request received 18
overview 17
sequencing rules
BATCHED_OFFER_MESSAGE request received 18
INITIAL_OFFER_MESSAGE request received 17
other message received 19
SEGMENT_INFO_MESSAGE request received 18
timer events 19
timers 17
Standards assignments 9
Syntax 10

T

Timer events
client 21
server 19
Timers
client 19
server 17
Tracking changes 28
Transport 10
Transport_Header packet 16
Triggered events - higher-layer
client 20
server 17

V

Vendor-extensible fields 9
Versioning 9