[MS-PCCRC]:
Peer Content Caching and Retrieval: Content Identification

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	0.1.2	Editorial	Revised and edited the technical content.
04/10/2009	0.2	Minor	Updated the technical content.
05/22/2009	1.0	Major	Updated and revised the technical content.
07/02/2009	1.1	Minor	Updated 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	1.6.3	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	1.6.3	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	1.6.3	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	1.6.3	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This document specifies the Content Information (section 2.3) format used by the Peer Content Caching and Retrieval Framework to uniquely identify **content** for discovery and retrieval purposes.

Peer Content Caching and Retrieval Framework is 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 available instead of the 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 framework also supports using hosted caches in place of peer-based caching. 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.

Content Information contains all the necessary information to allow a peer to uniquely specify content for discovery, and for peer caches or hosted caches to determine whether they have the specific content requested by the querying peer. Additionally, the Content Information should also ensure the confidentiality of content sent between peers and allow peers to verify the integrity of downloaded content **blocks**. To satisfy these requirements, Content Information utilizes cryptographic hashing and encryption algorithms to encrypt and generate hashes of the content units, and provides mechanisms to specify the starting point (offset) and length of the **content range**. Content Information is used in the Peer Content Caching and Retrieval Discovery Protocol and Retrieval Protocol to identify content for discovery, response, and retrieval requests.

1.1 Glossary

The following terms are specific to this document:

block: A subdivision of a **segment**. Each **segment** is divided into blocks of equal size (64 kilobytes (KB) in the current version) except for the last segment, which can be smaller if the **content** size is not a multiple of the standard **segment** sizes. Section 2.1 describes the relationship between **content**, **segment**, and blocks.

block hash: A hash of a content **block** within a **segment**. Also known as a block ID.

client: A **peer** that accesses certain **content**. It acts as a WS-Discovery client in the Discovery protocol, and in the client role in the Peer Content Caching and Retrieval: Retrieval Protocol and the Peer Content Caching and Retrieval: Hosted Cache protocols.

content: A file that an application accesses. Examples of content include Web pages and documents stored on either Web servers or SMB file servers.

content range: The starting offset and length for the **content** desired.

content server: A content server is the original server a **peer** contacts to obtain either the hashes of the **content** or the actual **content** when it is not available from the **peers**.

dataBlock: See **block**.

peer: The nodes participating in content caching and retrieval system. A peer is a node that both accesses the **content** and serves the content it caches for other peers.

Probe: The WS-Discovery protocol message sent by a **client** to discover **content**.

segment: A subdivision of **content**. Each segment has the same size (32 megabytes (MB) in the current version), except that the last segment can be smaller if the content size is not a multiple of the standard segment sizes. The relationship between **content**, segment, and **block** is described in section 2.1.

segment hash of data (HoD): The hash of the **content block hashes** of every **block** in the **segment**, regardless of how many of those **blocks** intersect the **content range**. The hash is of length 32 if **dwHashAlgo** at the start of the **content** information was 0x800C = SHA-256, 48 if **dwHashAlgo** = 0x800D = SHA-384 or 64 if **dwHashAlgo** = 0x800E = SHA-512.

segment ID (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**. The details of how a **segment ID** is generated are specified in section 2.2.

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 **segment hash of data** and the **server secret**.

server secret: A SHA-256 hash of an arbitrary length binary string stored on the server.

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.

[FIPS180-2] Federal Information Processing Standards Publication, "Secure Hash Standard", FIPS PUB 180-2, August 2002, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[FIPS197] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 197: Advanced Encryption Standard (AES)", November 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[MS-DTYP] Microsoft Corporation, "Windows Data Types", January 2007.

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

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

[RFC2104] Krawczyk, H., Bellare, M., and Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997, http://www.ietf.org/rfc/rfc2104.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

1.2.2 Informative References

None.

1.3 Overview

Content Information specifies a binary format used in the Peer Content Caching and Retrieval: Discovery Protocol, Peer Content Caching and Retrieval: Hosted Cache Protocol and Peer Content Caching and Retrieval: Retrieval Protocol of the Peer Content Caching and Retrieval Framework. For the detailed protocol operations, see the Peer Content Caching and Retrieval: Discovery Protocol [MS-PCCRD] and Peer Content Caching and Retrieval: Retrieval Protocol [MS-PCCRR] specifications.

Content Information is generated by a **content server** and supplied to clients requesting the content to allow them to take advantage of other peers' or a hosted cache's cached copies of content). The content is divided into large **segments** and subdivided into smaller blocks, and cryptographic hashes of these blocks and segments are used to identify and verify them, so that they can be retrieved correctly from peers instead of from the content server.

In order to ensure that cache content retrieval communications are at least as secure as a normal client's communications with a content server, all content servers must be configured with a binary secret value of arbitrary length. This secret is used as a key to derive secret keys to be used to secure communications between the peers or between peers and a Hosted Cache. If the secret value is not configured, it is automatically generated.

1.4 Relationship to Protocols and Other Structures

Content Information (section 2.3) is a binary data structure. Content Information contains all necessary information to allow peers to discover other peers with requested content, ensure the confidentiality of content sent between peers, and verify the integrity of downloaded content blocks. This includes a hash generated using a hash algorithm (which must be one of SHA-256, SHA-384, or SHA-512—see [FIPS180-2]), for each content block of a particular content segment, a hash (using the same algorithm) of all the content **block hashes** for a particular content segment (termed the **segment hash of data**), a hash (using the same algorithm) of the segment hash of data and a **server secret** (termed the **segment secret**), the start offset and length of the content range represented by the Content Information Data Structure, content offset of the start of each segment, length of each segment in bytes, size of a content block in each segment and hash algorithms used for segment related hashes (segment hash of data and segment secret) and content block hashes. Content Information can be transmitted in any way an application developer desires.

1.5 Applicability Statement

The Content Information Data Structure is applicable in a context where both content servers and peers support the protocols in the Peer Content Caching and Retrieval Framework.<1>

This framework is intended primarily for settings where content servers are available only over low-bandwidth and/or high-latency links, while peers are available over higher-bandwidth, lower latency links. The data structure is provided by content servers that support it to clients that support using it to obtain their requested content from peers instead of the server.

A requesting peer contacts the content server to obtain the Content Information for the specific document it wishes to download. The peer then embeds identifiers from the Content Information in the Discovery Protocol **probe** message (as specified in section 2.2.2.1) it sends out to the peer caches. A peer cache receiving the probe message takes these identifiers and replies to the requesting peer if they match the Content Information cached locally. The actual content retrieval thus uses primarily the higher-bandwidth, lower-latency links.

1.6 Versioning and Localization

This document covers versioning issues in the following areas:

- Structure Versions: The Content Information Data Structure (section 2.3) supports only version 1.0. The version is defined in section 2.

1.7 Vendor-Extensible Fields

There are no vendor-extensible fields in the Content Information Data Structure (section 2.3).

2 Structures

This section describes the Content Information Data Structure (section 2.3). Before defining the structure layout, the first two subsections describe the relationship between content, segments, and blocks, and also explain the procedures for generating various cryptographic hashes used to identify segments and blocks.

2.1 Content, Segments, and Blocks

For the purposes of the Peer Content Caching and Retrieval Framework, content is considered to be divided into one or more segments. Each segment is a binary string of a standard size (32 megabytes), except possibly the last segment which may be smaller if the content size is not a multiple of the standard segment size. Each segment is identified on the network by its segment ID (see section 2.2), also known as **HoHoDk**. Different content items can share the same segment if they happen to contain an identical part that coincides with a complete segment.

Each segment is divided in turn into blocks. Each block is a binary string of a fixed size, (64 kilobytes), except for the last block in the last segment, which again may be shorter. Unlike segments, blocks in different segments are always considered distinct objects, even if identical. Blocks within a segment are identified by their progressive index within the segment (Block 0 is the first block in the segment, Block 1 the second, …). Because of the fixed block size, a block's index can also be used to compute its actual byte offset in the segment. Given the standard block size of 64 kilobytes, Block 0 is located at offset 0 in the segment, Block 1 at offset 65536, Block 2 at offset 131072, etc.

Note that given the entire set of blocks for a segment, each identified by index, one can reconstruct the original segment simply by concatenating the blocks in order by index. Similarly, given the entire sequence of HoHoDk values for the successive segments in a content item, and a set of segments with matching associated HoHoDk values, one can reconstruct the original content simply by concatenating the segments in order based on HoHoDk value.

2.2 Segment Identifiers (HoHoDk) and Keys

Cryptographic hashes are used to identify segments and blocks. The following describes the procedures to generate a segment secret, a segment hash of data, and a segment ID (HoHoDk). These computations use the following inputs:

- The content.
- A hashing algorithm (configurable), used either directly or as part of the HMAC mechanism (see [RFC2104]). The hashing algorithm is assumed to take an arbitrary-length byte string as input, and to output a fixed-length binary string as output. The list of possible hashing algorithms can be found in section 2.3.
- A server-configured secret, in the form of a binary string.

The following diagram shows how the set of hashes is calculated:

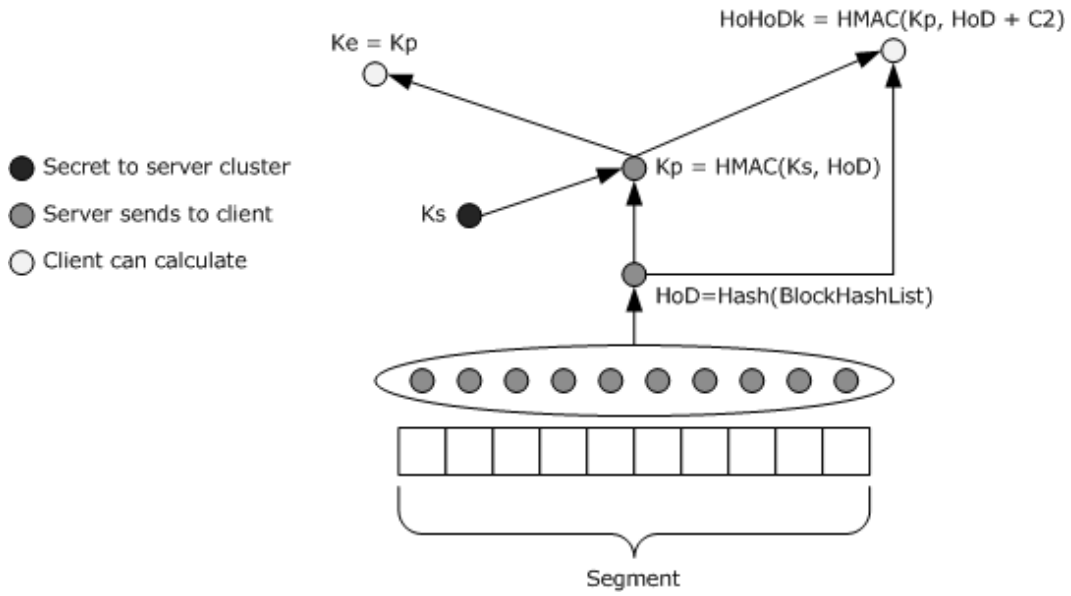


Figure 1: Calculation of Hashes

Notation:

- "+": concatenation
- Hash: The input hash function, which MUST be one of the hash functions listed in section 2.3.
- HMAC: The HMAC function (defined in [RFC2104]), constructed using Hash as the hash function.
- Ks: The server secret.
- Kp (segment secret): A segment-specific hash that is sent to authorized clients along with the rest of the content information.
- HoD (segment hash of data): The hash of the content block hashes of every block in the segment, as formulated below.
- HoHoDk (segment ID): 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. HoHoDk is disclosed freely in broadcast messages, as specified in [MS-PCCRD]. Knowledge of the HoHoDk does not prove authorization to access the data.
- Ke: An encryption key derived from the segment secret. A sending peer will encrypt data (using the mechanism described in [FIPS197]) with Ke but Ke is never disclosed between peers. The receiving client must already have obtained enough information to compute the value of Ke from a server in order to decrypt the peer-supplied data. Ke = Kp.
- ContentInfo: The segment-specific data that is sent to authorized clients as part of the Content Information Data Structure (section 2.3) for each segment of content they access. ContentInfo includes only the list of block hashes, the HoD, and Kp. Ke and HoHoDk are not included in ContentInfo. A peer receiving the ContentInfo derives them from the available values according to the formulae below. Ks is never disclosed by the server.
- **dataBlock**: dataBlock and block are used interchangeably.

Formulae:

segment = dataBlock1 + dataBlock2 + … + dataBlockn (where the segment consists of n blocks; see section 2.1 for details.)

BlockHashi = Hash(dataBlocki) 1<=i<=n

BlockHashList = BlockHash1 + BlockHash2 + … + BlockHashN

HoD = Hash(BlockHashList)

Kp = HMAC(Ks, HoD)

Ke = Kp

HoHoDk = HMAC(Kp, HoD +C2) (where C2 is the null-terminated ASCII string constant "MS_P2P_CACHING"; string literals are all ASCII strings with NULL terminators unless otherwise noted.)

ContentInfo = HoD + Kp + BlockHashList

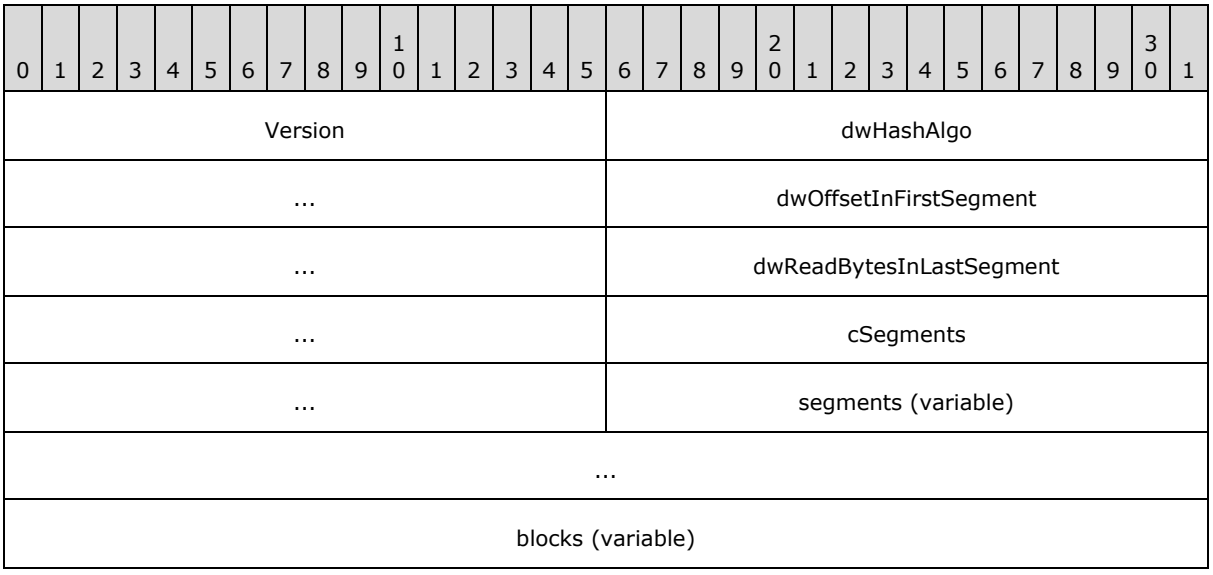
Before making ANY received blocks in a segment available to the higher layer, any peers or the hosted cache, a peer MUST verify that:

- Each block hash matches the supplied HoD.
- The hash of each dataBlock matches the corresponding supplied block hash.

2.3 Content Information Data Structure

Content Information is a variable size data structure. Content Information size is proportional to the length of the content it represents.

Content Information starts with a single 2 byte WORD value representing the data structure version. Version 1.0 of the Content Information data structure is formatted as follows. All fields are in host byte order.



…

Version (2 bytes): Content Information version (0x0100 is version 1.0). The low byte is the minor version number and the high byte is the major version number. MUST be 0x0100.

dwHashAlgo (4 bytes): Hash algorithm to use.<2> MUST be one of the following values:

Value	Meaning
0x0000800C	Use the SHA-256 hash algorithm.
0x0000800D	Use the SHA-384 hash algorithm.
0x0000800E	Use the SHA-512 hash algorithm.

dwOffsetInFirstSegment (4 bytes): Number of bytes into the first segment within the Content Information data structure at which the content range begins.

dwReadBytesInLastSegment (4 bytes): Total number of bytes of the content range which lie within the final segment in the Content Information data structure.

cSegments (4 bytes): The number of segments which intersect the content range and hence are contained in the Content Information data structure.

segments (variable): Segment start offset, length, block size, **SegmentHashofData** and **SegmentSecret** for each segment. Each segment description is as specified in 2.3.1.1.

blocks (variable): Count of blocks and content block hashes for each block intersecting the content range for each segment in the Content Information data structure. Each set of blocks for a segment is as specified in 2.3.1.2.

2.3.1 Fields

2.3.1.1 SegmentDescription

The **segments** field is composed of a number **cSegments** of SegmentDescription fields. Each SegmentDescription field corresponds to a content segment in the order in which they appear in the original content. Every segment except for the last segment must be exactly 32 megabytes in size. The Content Information Data Structure (section 2.3) defines the content range as described below.

Content range = {Start offset, Length}

Start offset = **ullOffsetInContent** + **dwOffsetInFirstSegment**, where **ullOffsetInContent** is taken from the first SegmentDescription in the **segments** field.

Length =(Sum of **cbSegment** of all segments in **segments** field except for the first segment and last segment) + (**cbSegment** of first segment – **dwOffsetInFirstSegment**) + **dwReadBytesInLastSegment**

The content range extends to the end of all the segments whose SegmentDescriptions are included in the Content Information except for the last segment, for which the number of bytes is limited to **dwReadBytesInLastSegment** instead of the total number of bytes actually present in the segment.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ullOffsetInContent																															
...																															
cbSegment																															
cbBlockSize																															
SegmentHashOfData (variable)																															
...																															
SegmentSecret (variable)																															
...																															

ullOffsetInContent (8 bytes): Content offset at which the start of the segment begins.

cbSegment (4 bytes): Total number of bytes in the segment, regardless of how many of those bytes intersect the content range.

cbBlockSize (4 bytes): Length of a content block within this segment, in bytes. Every segment MUST use the block size of 65536 bytes.

SegmentHashOfData (variable): The hash of the content block hashes of every block in the segment, regardless of how many of those blocks intersect the content range. The hash is of length 32 if **dwHashAlgo** at the start of the Content Information was 0x800C = SHA-256, 48 if **dwHashAlgo** = 0x800D = SHA-384 or 64 if **dwHashAlgo** = 0x800E = SHA-512.

SegmentSecret (variable): Kp (see section 2.2), computed as Hash (SegmentHashofData + ServerSecret) using the hash algorithm specified at the beginning of the Content Information Data Structure. The hash is of length 32 if **dwHashAlgo** at the start of the Content Information was 0x800C = SHA-256, 48 if **dwHashAlgo** = 0x800D = SHA-384 or 64 if **dwHashAlgo** = 0x800E = SHA-512.

2.3.1.2 SegmentContentBlocks

The blocks field contains a number **cSegments** of SegmentContentBlocks fields. The Nth SegmentContentBlocks field corresponds to the Nth SegmentDescription (section 2.3.1.1) and hence the Nth content segment. The SegmentContentBlocks field is formatted as follows.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cBlocks																															
BlockHashes (variable)																															

…

cBlocks (4 bytes): Number of content blocks in the segment which intersect the content range specified at the start of the Content Information (section 2.3).

BlockHashes (variable): SHA-256, SHA-384 or SHA-512 hash of each content block in the order in which the blocks appear in the segment. The size of this field is **cBlocks** * (32, 48 or 64, depending on which hash was used).

3 Structure Examples

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

3.1 125 KB Content

Scenario: A server *S* wants clients to use the Content Caching and Retrieval Framework to accelerate content distribution for a 125 kilobyte (KB) file. The server is configured to use SHA-256 as the hash algorithm and uses a secret value which is the ASCII string "no more secrets". A client requests the entirety of the 125 kilobyte content from the server. The server responds with Content Information of the following form.

Offset	Size	Type	Name	Value
0	2	WORD	Version	0x100
2	4	DWORD	dwHashAlgo	0x800C
6	4	DWORD	dwOffsetInFirstSegment	0
10	4	DWORD	dwReadBytesInLastSegment	128000
14	4	DWORD	cSegments	1
18	8	ULONGLONG	ullOffsetInContent	0
26	4	DWORD	cbSegment	128000
30	4	DWORD	cbBlockSize	0x10000
34	32	BYTE[]	SegmentHashofData	SHA-256 hash of the 2 content block hashes.
66	32	BYTE[]	SegmentSecret	SHA-256 hash of segment hash of data and "no more secrets".
98	4	DWORD	cBlocks	2
102	32	BYTE[]	block hash	SHA-256 hash of the first 64-kilobyte block of the content.
134	32	BYTE[]	block hash	SHA-256 hash of the last 62,464 bytes of the content.

3.2 125 MB Content

Scenario: The same server S now receives a client request for a 125-megabyte file. The server responds with Content Information of the following form.

Offset	Size	Type	Name	Value
0	2	WORD	Version	0x100
2	4	DWORD	dwHashAlgo	0x800C
6	4	DWORD	dwOffsetInFirstSegment	0

Offset	Size	Type	Name	Value
10	4	DWORD	dwReadBytesInLastSegment	0x1D00000
14	4	DWORD	cSegments	4
18	8	ULONGLONG	ullOffsetInContent	0
26	4	DWORD	cbSegment	0x2000000
30	4	DWORD	cbBlockSize	0x10000
34	32	BYTE[]	SegmentHashofData	SHA-256 hash of the all 512 content block hashes for the first segment.
66	32	BYTE[]	SegmentSecret	SHA-256 hash of segment hash of data and "no more secrets".
98	8	ULONGLONG	ullOffsetInContent	0x2000000
106	4	DWORD	cbSegment	0x2000000
110	4	DWORD	cbBlockSize	0x10000
114	32	BYTE[]	SegmentHashofData	SHA-256 hash of the all 512 content block hashes for the second segment.
146	32	BYTE[]	SegmentSecret	SHA-256 hash of segment hash of data and “no more secrets”.
178	8	ULONGLONG	ullOffsetInContent	0x4000000
186	4	DWORD	cbSegment	0x2000000
190	4	DWORD	cbBlockSize	0x10000
194	32	BYTE[]	SegmentHashofData	SHA-256 hash of the all 512 content block hashes for the third segment.
226	32	BYTE[]	SegmentSecret	SHA-256 hash of segment hash of data and “no more secrets”.
258	8	ULONGLONG	ullOffsetInContent	0x6000000
266	4	DWORD	cbSegment	0x1D00000
270	4	DWORD	cbBlockSize	0x10000
274	32	BYTE[]	SegmentHashofData	SHA-256 hash of the all 464 content block hashes for the fourth segment.
306	32	BYTE[]	SegmentSecret	SHA-256 hash of segment hash of data and “no more secrets”.
338	4	DWORD	cBlocks	512
342	32	BYTE[]	block hash	SHA-256 hash of the first 64 kilobyte block of the content.
…				

Offset	Size	Type	Name	Value
16694	32	BYTE[]	block hash	SHA-256 hash of bytes 0x1FF0000 through 0x2000000 of the content.
16726	4	DWORD	cBlocks	512
16730	32	BYTE[]	block hash	SHA-256 hash of bytes 0x2000000 through 0x2010000 of the content.
…				
33082	32	BYTE[]	block hash	SHA-256 hash of bytes 0x3FF0000 through 0x4000000 of the content.
33114	4	DWORD	cBlocks	512
33118	32	BYTE[]	block hash	SHA-256 hash of bytes 0x4000000 through 0x4010000 of the content.
…				
49470	32	BYTE[]	block hash	SHA-256 hash of bytes 0x5FF0000 through 0x6000000 of the content.
49502	4	DWORD	cBlocks	464
49506	32	BYTE[]	block hash	SHA-256 hash of bytes 0x6000000 through 0x6010000 of the content.
…				
64322	32	BYTE[]	block hash	SHA-256 hash of bytes 0x7CF0000 through 0x7D00000 of the content.

4 Security Considerations

4.1 Download Confidentiality

The Download Protocol transport is built upon HTTP and HTTP is a stateless protocol. Therefore, in order to secure communications between peers or between peers and a Hosted Cache any content data blocks sent in messages are encrypted using the segment secret of the content segment within which the content blocks are contained. This ensures that it is intractable for an entity which is not in possession of the server secret used to derive the segment secret to discover the actual data contained in such an encrypted data block.

The segment secret Kp must be treated with the same degree of security as the plaintext segment itself, since knowledge of it for a given segment is sufficient to obtain the segment from peers using the Peer Content Caching and Retrieval Framework protocols, and then decrypt it. Knowledge of Ks does not immediately yield any particular plain text, but can be used to glean certain types of data from the cipher text, and possibly expose some partially-known data to brute-force guessing attack. It therefore SHOULD be kept confidential.

4.2 Content Block Validation

Peers must validate that content blocks downloaded from other peers or a Hosted Cache contain the same data as the original content available from the server which supplied the Content Information (section 2.3). Peers accomplish this by hashing received content blocks using the block hash algorithm specified in the Content Information and comparing the hash with the content block hash specified for that particular content block in the Content Information. If the hashes match then the peer can be confident that the data matches the content available from the server.

5 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 1.5: 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 via Windows Management Framework. Support for the server-side elements of this protocol is not available for Windows Vista or Windows Server 2008.

<2> Section 2.3: In Windows 7, the [MS-PCCRR] implementation can only accept segment IDs generated using SHA-256, whereas the Windows Server 2008 R2 implementation of the hash generation part can generate segment IDs using one of three hashing algorithms: SHA-256, SHA-384, or SHA-512.

6 Change Tracking

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

7 Index