Network Working Group
Request for Comments: 2756
Category: Experimental

P. Vixie
ISC
D. Wessels
NLANR

January 2000

Hyper Text Caching Protocol (HTCP/0.0)

Status of this Memo

Copyright Notice

Abstract

This document describes HTCP, a protocol for discovering HTTP caches and cached data, managing sets of HTTP caches, and monitoring cache activity. This is an experimental protocol, one among several proposals to perform these functions.

1. Definitions, Rationale and Scope

1.1. HTTP/1.1 (see [RFC2616]) permits the transfer of web objects from "origin servers," possibly via "proxies" (which are allowed under some circumstances to "cache" such objects for subsequent reuse) to "clients" which consume the object in some way, usually by displaying it as part of a "web page." HTTP/1.0 and later permit "headers" to be included in a request and/or a response, thus expanding upon the HTTP/0.9 (and earlier) behaviour of specifying only a URI in the request and offering only a body in the response.

1.2. ICP (see [RFC2186]) permits caches to be queried as to their content, usually by other caches who are hoping to avoid an expensive fetch from a distant origin server. ICP was designed with HTTP/0.9 in mind, such that only the URI (without any headers) is used when describing cached content, and the possibility of multiple compatible bodies for the same URI had not yet been imagined.

1.3. This document specifies a Hyper Text Caching Protocol (HTCP) which permits full request and response headers to be used in cache management, and expands the domain of cache management to include monitoring a remote cache's additions and deletions, requesting immediate deletions, and sending hints about web objects such as the third party locations of cacheable objects or the measured uncacheability or unavailability of web objects.

2. HTCP Protocol

2.1. All multi-octet HTCP protocol elements are transmitted in network byte order. All RESERVED fields should be set to binary zero by senders and left unexamined by receivers. Headers must be presented with the CRLF line termination, as in HTTP.

2.2. Any hostnames specified should be compatible between sender and receiver, such that if a private naming scheme (such as HOSTS.TXT or NIS) is in use, names depending on such schemes will only be sent to HTCP neighbors who are known to participate in said schemes. Raw addresses (dotted quad IPv4, or colon-format IPv6) are universal, as are public DNS names. Use of private names or addresses will require special operational care.

2.3. HTCP messages may be sent as UDP datagrams, or over TCP connections. UDP must be supported. HTCP agents must not be isolated from NETWORK failures and delays. An HTCP agent should be prepared to act in useful ways when no response is forthcoming, or when responses are delayed or reordered or damaged. TCP is optional and is expected to be used only for protocol debugging. The IANA has assigned port 4827 as the standard TCP and UDP port number for HTCP.

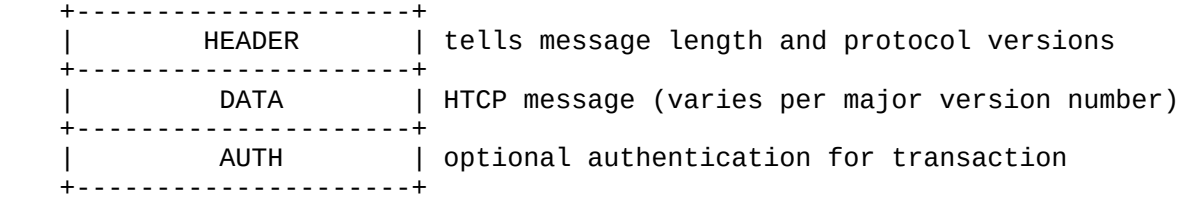
2.4. A set of configuration variables concerning transport characteristics should be maintained for each agent which is capable of initiating HTCP transactions, perhaps with a set of per-agent global defaults. These variables are:

Maximum number of unacknowledged transactions before a "failure" is imputed.

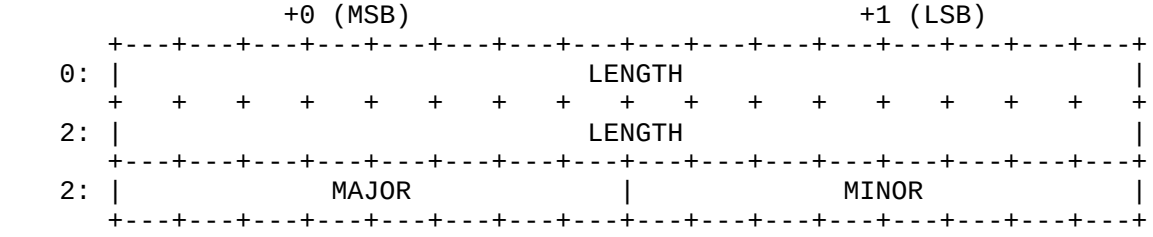
Maximum interval without a response to some transaction before a "failure" is imputed.

Minimum interval before trying a new transaction after a failure.

2.5. An HTCP Message has the following general format:



2.6. An HTCP/*.* HEADER has the following format:



LENGTH is the message length, inclusive of all header and data octets, including the LENGTH field itself. This field will be equal to the datagram payload size ("record length") if a datagram protocol is in use, and can include padding, i.e., not all octets of the message need be used in the DATA and AUTH sections.

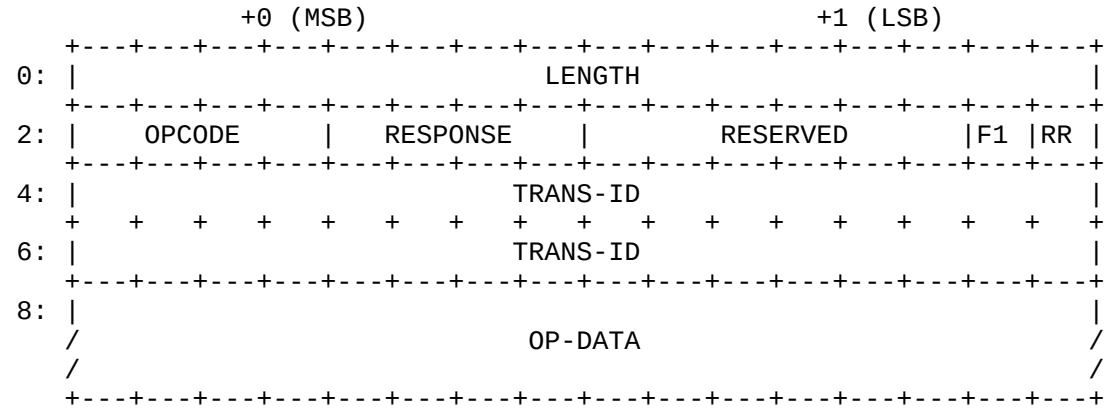
MAJOR is the major version number (0 for this specification). The DATA section of an HTCP message need not be upward or downward compatible between different major version numbers.

MINOR is the minor version number (0 for this specification). Feature levels and interpretation rules can vary depending on this field, in particular RESERVED fields can take on new (though optional) meaning in successive minor version numbers within the same major version number.

2.6.1. It is expected that an HTCP initiator will know the version number of a prospective HTCP responder, or that the initiator will probe using declining values for MINOR and MAJOR (beginning with the highest locally supported value) and locally cache the probed version number of the responder.

2.6.2. Higher MAJOR numbers are to be preferred, as are higher MINOR numbers within a particular MAJOR number.

2.7. An HTCP/0.* DATA has the following structure:



LENGTH is the number of octets of the message which are reserved for the DATA section, including the LENGTH field itself. This number can include padding, i.e., not all octets reserved by LENGTH need be used in OP-DATA.

OPCODE is the operation code of an HTCP transaction. An HTCP transaction can consist of multiple HTCP messages, e.g., a request (sent by the initiator), or a response (sent by the responder).

RESPONSE is a numeric code indicating the success or failure of a transaction. It should be set to zero (0) by requestors and ignored by responders. Each operation has its own set of response codes, which are described later. The overall message has a set of response codes which are as follows:

0 authentication wasn't used but is required
1 authentication was used but unsatisfactorily
2 opcode not implemented
3 major version not supported
4 minor version not supported (major version is ok)
5 inappropriate, disallowed, or undesirable opcode

The above response codes all indicate errors and all depend for their visibility on MO=1 (as specified below).

RR is a flag indicating whether this message is a request (0) or response (1).

F1 is overloaded such that it is used differently by requestors than by responders. If RR=0, then F1 is defined as RD. If RR=1, then F1 is defined as MO.

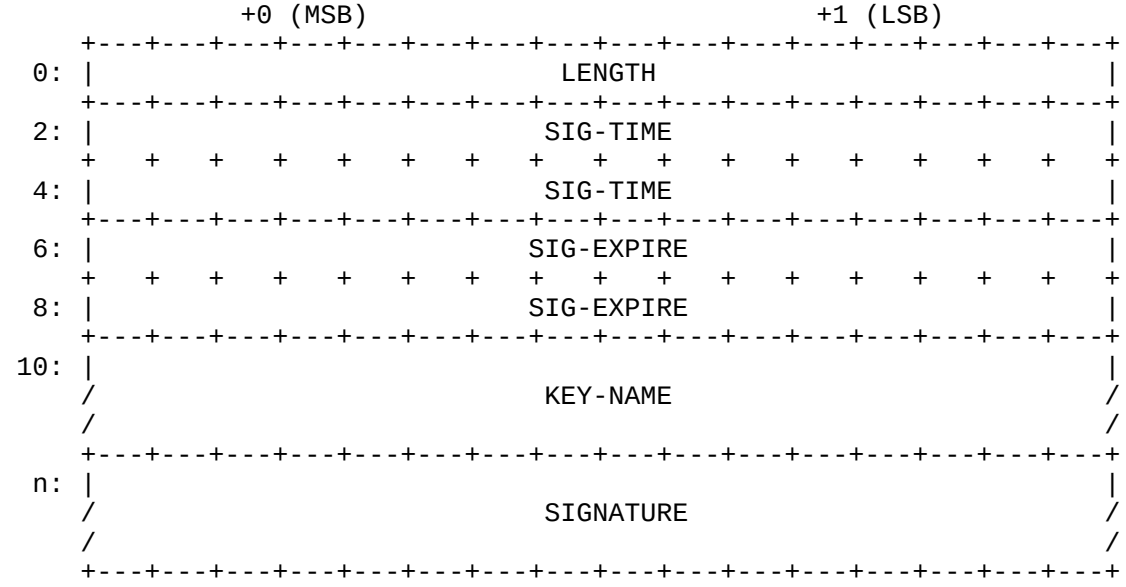
RD is a flag which if set to 1 means that a response is desired. Some OPCODEs require RD to be set to 1 to be meaningful.

MO (em-oh) is a flag which indicates whether the RESPONSE code is to be interpreted as a response to the overall message (fixed fields in DATA or any field of AUTH) [MO=1] or as a response to fields in the OP-DATA [MO=0].

TRANS-ID is a 32-bit value which when combined with the initiator's network address, uniquely identifies this HTCP transaction. Care should be taken not to reuse TRANS-ID's within the life-time of a UDP datagram.

OP-DATA is opcode-dependent and is defined below, per opcode.

2.8. An HTCP/0.0 AUTH has the following structure:



LENGTH is the number of octets used by the AUTH, including the LENGTH field itself. If the optional AUTH is not being transmitted, this field should be set to 2 (two). LENGTH can include padding, which means that not all octets reserved by LENGTH will necessarily be consumed by SIGNATURE.

SIG-TIME is an unsigned binary count of the number of seconds since 00:00:00 1-Jan-70 UTC at the time the SIGNATURE is generated.

SIG-EXPIRE is an unsigned binary count of the number of seconds since 00:00:00 1-Jan-70 UTC at the time the SIGNATURE is considered to have expired.

KEY-NAME is a COUNTSTR [3.1] which specifies the name of a shared secret. (Each HTCP implementation is expected to allow configuration of several shared secrets, each of which will have a name.)

SIGNATURE is a COUNTSTR [3.1] which holds the HMAC-MD5 digest (see [RFC 2104]), with a B value of 64, of the following elements, each of which is digested in its "on the wire" format, including transmitted padding if any is covered by a field's associated LENGTH:

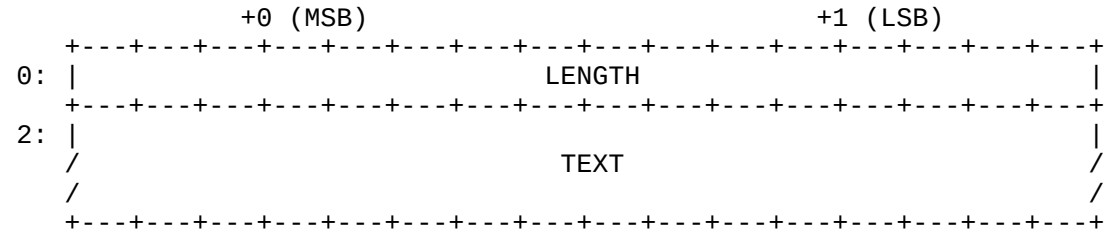
IP SRC ADDR	[4 octets]
IP SRC PORT	[2 octets]
IP DST ADDR	[4 octets]
IP DST PORT	[2 octets]
HTCP MAJOR version number	[1 octet]
HTCP MINOR version number	[1 octet]
SIG-TIME	[4 octets]
SIG-EXPIRE	[4 octets]
HTCP DATA	[variable]
KEY-NAME (the whole COUNTSTR [3.1])	[variable]

2.8.1. Shared secrets should be cryptorandomly generated and should be at least a few hundred octets in size.

3. Data Types

HTCP/0.* data types are defined as follows:

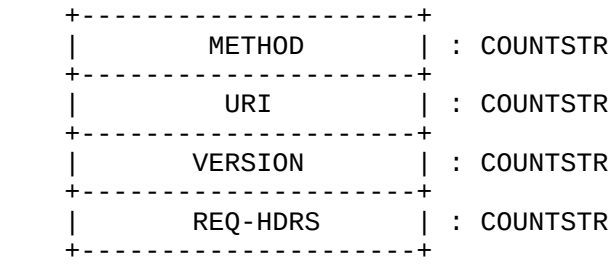
3.1. COUNTSTR is a counted string whose format is:



LENGTH is the number of octets which will follow in TEXT. This field is *not* self-inclusive as is the case with other HTCP LENGTH fields.

TEXT is a stream of uninterpreted octets, usually ISO8859-1 "characters".

3.2. SPECIFIER is used with the TST and CLR request messages, defined below. Its format is:



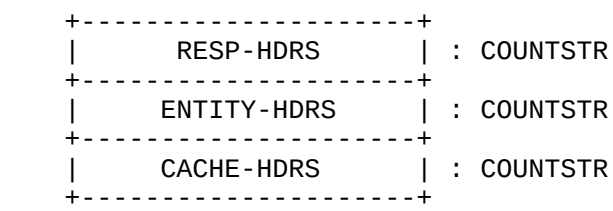
METHOD (Since HTCP only returns headers, methods GET and HEAD are equivalent.)

URI (If the URI is a URL, it should always include a ":"<port> specifier, but in its absense, port 80 should be imputed by a receiver.)

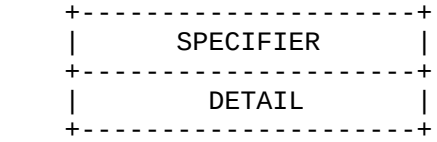
VERSION is an entire HTTP version string such as" HTTP/1.1". VERSION strings with prefixes other than "HTTP/" or with version numbers less than "1.1" are outside the domain of this specification.

REQ-HDRS are those presented by an HTTP initiator. These headers should include end-to-end but NOT hop-by-hop headers, and they can be canonicalized (aggregation of "Accept:" is permitted, for example.)

3.3. DETAIL is used with the TST response message, defined below. Its format is:



3.4. IDENTITY is used with the MON request and SET response message, defined below. Its format is:



4. Cache Headers

HTCP/0.0 CACHE-HDRS consist of zero or more of the following headers:

Cache-Vary: <header-name> ...
The sender of this header has learned that content varies on a set of headers different from the set given in the object's Vary: header. Cache-Vary:, if present, overrides the object's Vary: header.

Cache-Location: <cache-hostname>:<port> ...
The sender of this header has learned of one or more proxy caches who are holding a copy of this object. Probing these caches with HTCP may result in discovery of new, close-by (preferrable to current) HTCP neighbors.

Cache-Policy: [no-cache] [no-share] [no-cache-cookie]
The sender of this header has learned that the object's caching policy has more detail than is given in its response headers.

no-cache	means that it is uncacheable (no reason given), but may be shareable between simultaneous requestors.
no-share	means that it is unshareable (no reason given), and per-requestor tunnelling is always required).

no-cache-cookie means that the content could change as a result of different, missing, or even random cookies being included in the request headers, and that caching is inadvisable.

Cache-Flags: [incomplete]
The sender of this header has modified the object's caching policy locally, such that requesters may need to treat this response specially, i.e., not necessarily in accordance with the object's actual policy.

incomplete means that the response headers and/or entity headers given in this response are not known to be complete, and may not be suitable for use as a cache key.

Cache-Expiry: <date>
The sender of this header has learned that this object should be considered to have expired at a time different than that indicated by its response headers. The format is the same as HTTP/1.1 Expires:.

Cache-MD5: <discovered content MD5>
The sender of this header has computed an MD5 checksum for this object which is either different from that given in the object's Content-MD5: header, or is being supplied since the object has no Content-MD5 header. The format is the same as HTTP/1.1 Content-MD5:.

Cache-to-Origin: <origin> <rtt> <samples> <hops>
The sender of this header has measured the round trip time to an origin server (given as a hostname or literal address). The <rtt> is the average number of seconds, expressed as decimal ASCII with arbitrary precision and no exponent. <Samples> is the number of RTT samples which have had input to this average. <Hops> is the number of routers between the cache and the origin, expressed as decimal ASCII with arbitrary precision and no exponent, or 0 if the cache doesn't know.

6. HTCP Operations

HTCP/0.* opcodes and their respective OP-DATA are defined below:

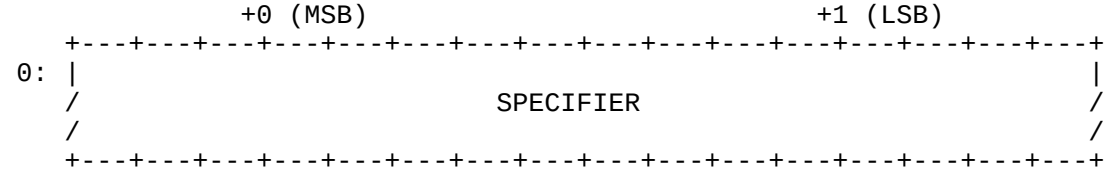
6.1. NOP (OPCODE 0):

This is an HTCP-level "ping." Responders are encouraged to process NOP's with minimum delay since the requestor may be using the NOP RTT (round trip time) for configuration or mapping purposes. The RESPONSE code for a NOP is always zero (0). There is no OP-DATA for a NOP. NOP requests with RD=0 cause no processing to occur at all.

6.2. TST (OPCODE 1):

Test for the presence of a specified content entity in a proxy cache. TST requests with RD=0 cause no processing to occur at all.

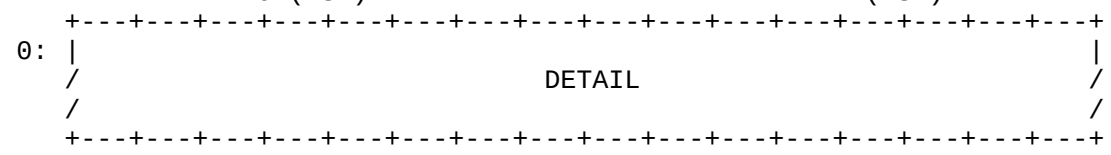
TST requests have the following OP-DATA:



RESPONSE codes for TST are as follows:

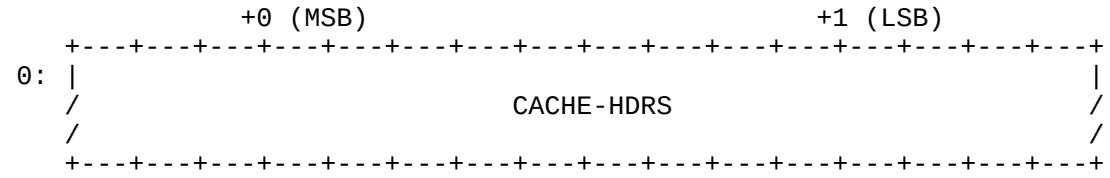
0 entity is present in responder's cache
1 entity is not present in responder's cache

TST responses have the following OP-DATA, if RESPONSE is zero (0):



Note: The response headers returned by a positive TST can be of a stale object. Requestors should be prepared to cope with this condition, either by using the responder as a source for this object (which could cause the responder to simply refresh it) or by choosing a different responder.

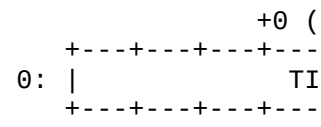
TST responses have the following OP-DATA, if RESPONSE is one (1):



6.3. MON (OPCODE 2):

Monitor activity in a proxy cache's local object store (adds, deletes, replacements, etc). Since interleaving of HTCP transactions over a single pair of UDP endpoints is not supported, it is recommended that a unique UDP endpoint be allocated by the requestor for each concurrent MON request. MON requests with RD=0 are equivalent to those with RD=1 and TIME=0; that is, they will cancel any outstanding MON transaction.

MON requests have the following OP-DATA structure:

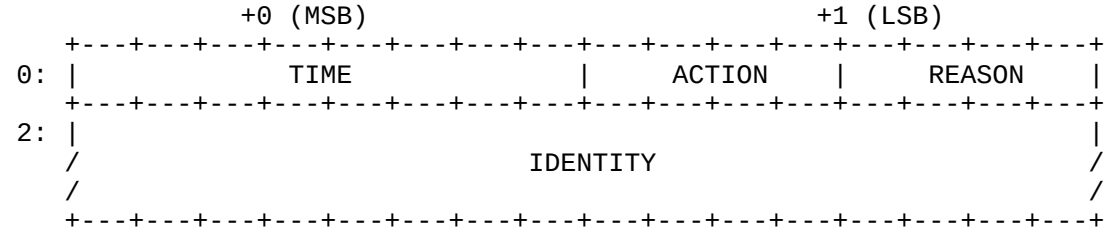


TIME is the number of seconds of monitoring output desired by the initiator. Subsequent MON requests from the same initiator with the same TRANS-ID should update the time on a ongoing MON transaction. This is called "overlapping renew."

RESPONSE codes for MON are as follows:

0 accepted, OP-DATA is present and valid
1 refused (quota error -- too many MON's are active)

MON responses have the following OP-DATA structure, if RESPONSE is zero (0):



TIME is the number of seconds remaining for this MON transaction.

ACTION is a numeric code indicating a cache population action. Codes are:

0 an entity has been added to the cache
1 an entity in the cache has been refreshed
2 an entity in the cache has been replaced
3 an entity in the cache has been deleted

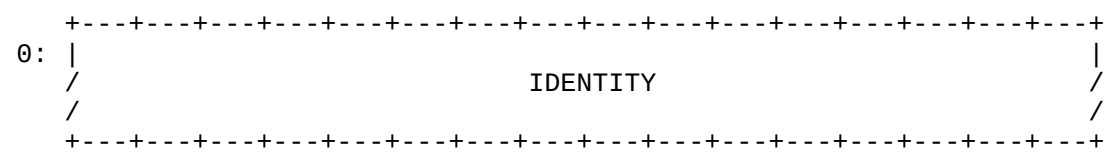
REASON is a numeric code indicating the reason for an ACTION. Codes are:

0 some reason not covered by the other REASON codes
1 a proxy client fetched this entity
2 a proxy client fetched with caching disallowed
3 the proxy server prefetched this entity
4 the entity expired, per its headers
5 the entity was purged due to caching storage limits

6.4. SET (OPCODE 3):

Inform a cache of the identity of an object. This is a "push" transaction, whereby cooperating caches can share information such as updated Age/Date/Expires headers (which might result from an origin "304 Not modified" HTTP response) or updated cache headers (which might result from the discovery of non-authoritative "vary" conditions or from learning of second or third party cache locations for this entity. RD is honoured.

SET requests have the following OP-DATA structure:



RESPONSE codes are as follows:

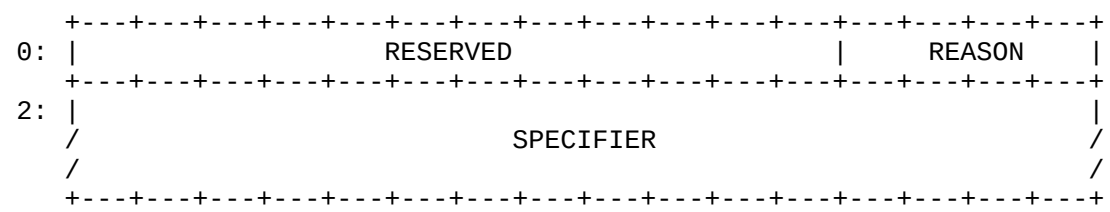
0 identity accepted, thank you
1 identity ignored, no reason given, thank you

SET responses have no OP-DATA.

6.5. CLR (OPCODE 4):

Tell a cache to completely forget about an entity. RD is honoured.

CLR requests have the following OP-DATA structure:



REASON is a numeric code indicating the reason why the requestor is asking that this entity be removed. The codes are as follows:

- 0 some reason not better specified by another code
- 1 the origin server told me that this entity does not exist

RESPONSE codes are as follows:

- 0 i had it, it’s gone now
- 1 i had it, i’m keeping it, no reason given.
- 2 i didn’t have it

CLR responses have no OP-DATA.

Clearing a URI without specifying response, entity, or cache headers means to clear all entities using that URI.

7. Security Considerations

If the optional AUTH element is not used, it is possible for unauthorized third parties to both view and modify a cache using the HTCP protocol.

8. Acknowledgements

Mattias Wingstedt of Idonex brought key insights to the development of this protocol. David Hankins helped clarify this document.

9. References

[RFC2396] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2104] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February, 1997.

[RFC2186] Wessels, D. and K. Claffy, "Internet Cache Protocol (ICP), version 2", RFC 2186, September 1997.

10. Authors' Addresses

Paul Vixie
Internet Software Consortium
950 Charter Street
Redwood City, CA 94063

Phone: +1 650 779 7001
EMail: vixie@isc.org

Duane Wessels
National Lab for Applied Network Research
USCD, 9500 Gilman Drive
La Jolla, CA 92093

Phone: +1 303 497 1822
EMail: wessels@nlanr.net

11. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.