HYBI	S. Loreto	
Internet-Draft	Ericsson	
Intended status: Standards Track	M. Thomson	
Expires: December 11, 2010	Andrew Corporation	
	G. Wilkins	
	Webtide	
	June 9, 2010	

TOC

Hypertext Transfer Protocol (HTTP) Timeouts
draft-loreto-http-timeout-00

Abstract

A Timeout header is defined for Hypertext Transfer Protocol (HTTP). This end-to-end header informs an origin server and any intermediaries of the maximum time that a client will await a response to its request. A server can use this header to ensure that a timely response is generated. This also identifies requests as being potentially long-lived, and allows for better resource allocation for these requests. A Connection-Timeout header is defined for HTTP. This hop-by-hop header informs the entity at the other end of a connection of the maximum time that an idle connection is kept open. This header improves reliability by providing better information about the idle connection management policy of HTTP hosts.
Comments and feedback for this draft should be directed to the authors and the Apps Discuss list (discuss@apps.ietf.org).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress.”
This Internet-Draft will expire on December 11, 2010.

Table of Contents

---

1. Introduction

TOC

This document describes two headers that provide Hypertext Transfer Protocol (HTTP) (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” June 1999.) [RFC2616] hosts with timing information.
The Timeout header describes the time available for a single HTTP request.

The Connection-Timeout header describes the time that an idle connection is retained.

1.1. Request Timeouts

TOC

HTTP is a client-driven protocol. As such, a server is only able to provide information to a client when that client requests it. To address the need for servers to provide low-latency notifications to clients, a practice of long-polling (Loreto, S., Saint-Andre, P., Salsano, S., and G. Wilkins, “Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP,” February 2010.) [I-D.loreto-http-bidirectional] is commonplace.

Long-polling requires that a client maintain an open request to a server. The server withholds responses to this request until the event of interest occurs. The open request provides the server with a way to send a message immediately after the event occurs. After the response containing the notification is received, the client immediately opens another open request.

Long-polling differs significantly to many HTTP requests, which do not have significant delay between request and response. Long-polling requests monopolize a connection for the time that they are outstanding. For this reason, it is useful for a server or intermediary to identify a long-polling request. Identifying a long-polling request would enable specialized resource management for long-polling requests; for instance, separate connection allocations and management, longer timeout periods and long-poll optimizations.

A connection that is used for long-polling also appears to be idle for a significant period. Management of idle connections can interfere with the operation of a long-poll. Many intermediaries legitimately use timeouts to avoid clients from monopolizing the use of outgoing connections. If an intermediary times out a request, the server is unable to send a response.

Knowing the time available to provide a response can avoid problems with timeouts. Current implementations select times between 30 and 120 seconds, times that have been empirically determined to be safe. A server provides a null response before this time to ensure that the connection appears to be active to intermediaries. Such low values are potentially wasteful, since each null response and subsequent request generates traffic. A null response can also increase the latency of notifications, since the response removes the channel that the server would use for notifications.

The Timeout header defined in Section 3 (Timeout Header) describes the maximum time available to a server in providing a response to any particular request.

For the Timeout header, Section 3.1 (Existing Intermediaries) includes a discussion of the interaction with existing HTTP intermediaries and other intermediaries like network address translation (NAT) device.
[[Open: Any value in echoing the actual value back to a client? MT: inclined to say no: we need a _use_, other than pure information, there's already a mechanism for ensuring that server gets the right value, having the client update its value is of no great benefit, especially since that encourages use of dodgy heuristics (i.e., is this request going to the same place?)]]

1.2. Idle Connection Timeouts

TOC

Management of idle HTTP connections has an impact on long-lived communications between hosts. Hosts are able to close idle connections in order to reduce resource consumption.
Many clients choose not to send non-idempotent requests on idle connections. If the intermediary or server holding the other end of the connection chooses to close the connection while a non-idempotent request is in transit, the client has no way to tell if the request has succeeded. For this reason, many clients establish a new connection for every non-idempotent request. This is inefficient if the existing connection is usable connection: establishing a new connection adds significantly to the latency of the request.
Polling of resource state can more efficiently use connection resources when an idle connection timeout is known. A client that polls on an interval that is close to the connection timeout can ensure that polling requests are made before the idle connection is closed, avoiding any need to retry a request due to the aforementioned problem. Alternatively, a client that knows that the connection timeout is less than the polling interval can close the connection immediately after a poll, releasing resources.
A Connection-Timeout header is defined in Section 4 (Connection-Timeout Header). This hop-by-hop header informs hosts of the minumum time that a connection remains idle before it is closed. Indicating this time explicitly gives a client the knowledge necessary to reuse an idle connection with a greater assurance that the connection is usable.

---

2. Terminology

TOC

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 (Bradner, S., “Key words for use in RFCs to

Indicate Requirement Levels," March 1997.) [RFC2119] and indicate requirement levels for compliant implementations.

---

3. Timeout Header

TOC

The Timeout header is a end-to-end request header that indicates the maximum time that a client is prepared to await a response.

```
Timeout            = "Timeout" ":" timeout-value
timeout-value      = 1*DIGIT
```

A client adds a Timeout header to any request for which they are prepared to await a response. The client sets the header to the maximum time that they are prepared to wait.
The value of the Timeout header is a single integer value in seconds. An origin server interprets this header as the time between receipt of a complete request and the time that it generates and begins sending the response. A client will observe a longer time interval between request and response, as network transit and processing by intermediaries add delays. If this time is critical, a client SHOULD allow for delays in setting a value for the header.
An origin server MAY apply a lower value to the timeout based on local policy. An origin server MAY choose to take longer to produce a response, at the risk that the client is no longer able to use the response.
An HTTP intermediary MAY reduce the value of a Timeout header based on local policy. An intermediary MAY add a Timeout header if none is present. The value in the Timeout header MUST NOT be increased or removed.
The client MUST NOT set the value of the Timeout header greater than the value of the Connection-Timeout header.

3.1. Existing Intermediaries

TOC

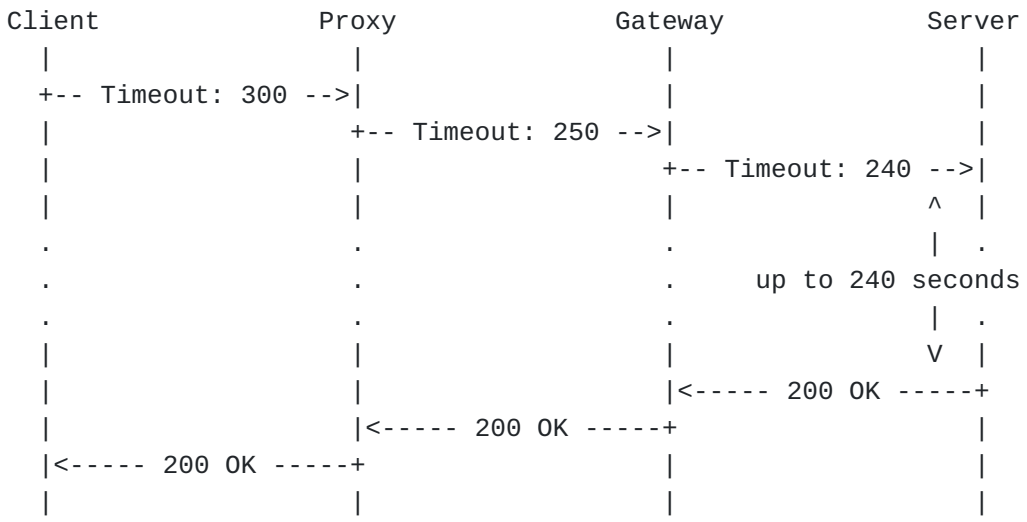
The exact impact of an intermediary on an HTTP request with a Timeout header depends on the type of intermediary.
An intermediary that is compliant with HTTP/1.1 passes this header, but unaware of the header semantics, passes the header on to the origin server without altering it. For an intermediary that does not time out requests, or an intermediary that has a timeout period that is longer than the one specified in the header, this has no impact.
An existing intermediary that has a shorter timeout period than the indicated time will time out a request before the server generates a

response. The intermediary could send the client a 504 (Gateway Timeout) response when it times out.
An intermediary that forwards requests without altering them (a transparent intermediary) is similar to an HTTP/1.1 compliant intermediary.
A non-compliant intermediary might remove a Timeout header. This means that the server is unaware of the timeliness requirements of the client. A server that expects a Timeout header can attempt to guess an appropriate value.
A network address translation (NAT) device might interact with timeouts. These devices maintain state about TCP connections that are established through them. This state can be discarded or lost if the connection remains idle. A NAT device failure can also cause state loss. Further attempts to use the connection either fail without error, or result in the NAT device sending a TCP RST to the entity that attempts to use the connection.
A client MAY revise the Timeout header that it sends in subsequent requests to the same resource or origin server if it detects intermediaries or NAT devices that have shorter timeout periods. However, two requests are not guaranteed to follow the same path through a network, especially for different resources. A client that employs a heuristic for setting the Timeout header SHOULD ensure that any knowledge it obtains about timeouts is not retained indefinitely.

3.2. Examples

TOC

The following example shows how a Timeout header could be used. In this case, both intermediaries - proxy and gateway - both reduce the timeout based on their policy.



---

4. Connection-Timeout Header

TOC

The Connection-Timeout header is a hop-by-hop header that indicates the minimum time that an idle connection will be maintained.

Connection-Timeout = "Connection-Timeout" ":" *timeout-value*
; timeout-value from Timeout header

This header is sent by either host participating in a persistent connection. A host sets the value to the minimum time that local policy at that host allows for an idle connection to remain open. A connection is idle if no data is sent or received by a host.
The value of the Connection-Timeout header is a single integer in seconds.
As a hop-by-hop header, this header only applies to a single transport-level connection. If a Connection-Timeout header is added to a request or response, the Connection header MUST include the tag Connection-Timeout. This ensures that intermediaries that do not recognize this header remove it before forwarding a request.
A host MAY keep an idle connection open for longer than the time that it indicates, but it SHOULD attempt to retain a connection for at least as long as indicated.
Network transit and processing by intermediaries add additional delays that can skew the subjective perception of whether a connection is idle. A client SHOULD make allowances for any delays in determining whether to reuse an idle connection.
An intermediary that provides a Connection-Timeout header value, MUST set the value of the Timeout header in any requests that it forwards to be less than or equal to the Connection-Timeout that it provides.

4.1. Existing Intermediaries

TOC

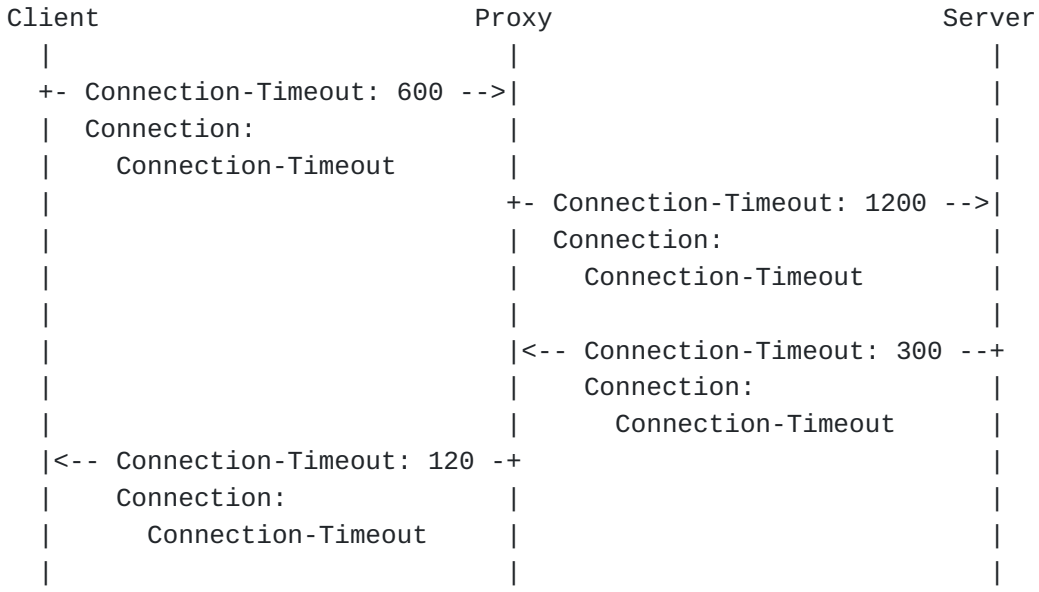
The exact impact of an intermediary on an HTTP request with a Connection-Timeout header depends on the type of intermediary.
An intermediary that is compliant with HTTP/1.1 ignores and discards this header before forwarding a request. Since it is unaware of the semantics of the header it could drop an idle connection at any time (see Section 7.1.4 of [I-D.ietf-httpbis-p1-messaging] (Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., and J. Reschke, “HTTP/1.1, part 1: URIs, Connections, and Message Parsing,” March 2010.)).
A non-compliant or otherwise transparent intermediary might pass this header on to the next hop. This results in same types of errors that the Keep-Alive header causes, as described in Section 19.7.1 of [RFC2068] (Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” January 1997.).

A network address translation (NAT) device might cause a connection to become unavailable prior to the advertised timeout.
A host MAY revise the Connection-Timeout header that it sends in subsequent requests to the same resource or origin server if it detects intermediaries or NAT devices that have shorter timeout periods.

4.2. Example

TOC

The following example shows how a Connection-Timeout header could be used. All connections are independently negotiated. In this example, the client indicates a timeout of 600 seconds (10 minutes), but the proxy is only prepared to retain the connection for at least 120 seconds (2 minutes). On the link between proxy and server, the proxy requests a timeout of 1200 seconds and the server reduces this to 300 seconds.



---

5. Security Considerations

TOC

Establishing a persistent connection requires a commitment of resources at a host. The Timeout and Connection-Timeout headers are used to express host policy that could alter the way that a host allocates connection resources.
A host that alters its policies based on the receipt of either header could be exploited by a malicious host to either benefit from increased access to resources or to consume resources in the hopes of depriving another host.

An attacker might attempt to keep a connection from becoming idle, to keep the connection open. A client could legitimately do this to benefit from a reduced latency for later requests, but an attacker might exploit this to monopolize server resources. A host that has indicated a Connection-Timeout MAY close a non-idle connection sooner than the indicated time if necessary or dictated by local policy (see Section 7.1.4 of [I-D.ietf-httpbis-p1-messaging] (Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., and J. Reschke, "HTTP/1.1, part 1: URIs, Connections, and Message Parsing," March 2010.)).

---

6. IANA Considerations

TOC

This section registers the two HTTP headers in the "Permanent Message Header Fields" registry established by [RFC3864] (Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields," September 2004.).
[[Note to IANA/RFC Editor: Please replace instance of RFCXXXX with the number of the published RFC and remove this note.]]

6.1. Timeout HTTP header Registration

TOC

This document registers the HTTP Timeout header in the "Permanent Message Header Fields" registry established by [RFC3864] (Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields," September 2004.)

> **Header field:** Timeout
>
> **Applicable protocol:** HTTP
>
> **Status:** standard
>
> **Author/change controller:** Internet Engineering Task Force, IETF (iesg@ietf.org)
>
> **Specification document(s):** RFCXXXX (this document)

TOC

6.2. Connection-Timeout HTTP header Registration

This document registers the HTTP Connection-Timeout header in the "Permanent Message Header Fields" registry established by [RFC3864] (Klyne, G., Nottingham, M., and J. Mogul, “Registration Procedures for Message Header Fields,” September 2004.)

> **Header field:** Connection-Timeout
>
> **Applicable protocol:** HTTP
>
> **Status:** standard
>
> **Author/change controller:** Internet Engineering Task Force, IETF (iesg@ietf.org)
>
> **Specification document(s):** RFCXXXX (this document)

---

7. References

TOC

7.1. Normative References

TOC

[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[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 (TXT, PS, PDF, HTML, XML).
[RFC3864]	Klyne, G., Nottingham, M., and J. Mogul, “Registration Procedures for Message Header Fields,” BCP 90, RFC 3864, September 2004 (TXT).

7.2. Normative References

TOC

[I-D.ietf-httpbis-p1-messaging]	Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., and J. Reschke, “HTTP/1.1, part 1: URIs, Connections, and Message Parsing,” draft-ietf-httpbis-p1-messaging-09 (work in progress), March 2010 (TXT).

[I-D.loreto-http-bidirectional]	Loreto, S., Saint-Andre, P., Salsano, S., and G. Wilkins, “Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP,” draft-loreto-http-bidirectional-02 (work in progress), February 2010 (TXT).
[RFC2068]	Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” RFC 2068, January 1997 (TXT).

Authors' Addresses

TOC

	Salvatore Loreto
	Ericsson
	Hirsalantie 11
	Jorvas 02420
	Finland
Email:	salvatore.loreto@ericsson.com
	Martin Thomson
	Andrew Corporation
	Andrew Building (39)
	Wollongong University Campus
	Northfields Avenue
	Wollongong, NSW 2522
	AU
Phone:	+61 2 4221 2915
Email:	martin.thomson@andrew.com
	Greg Wilkins
	Webtide
Email:	gregw@webtide.com