HYBI
Internet-Draft
Intended status: Standards Track
Expires: January 18, 2013

M. Thomson
Microsoft
S. Loreto
Ericsson
G. Wilkins
Intalio

July 17, 2012

Hypertext Transfer Protocol (HTTP) Keep-Alive Header
draft-thomson-hybi-http-timeout-03

Abstract

A Keep-Alive header is defined for HTTP. This hop-by-hop header informs hosts about connection management policies. A parameters is defined for idle connection timeout.

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 January 18, 2013.

Table of Contents

1. Introduction

This document describes the "Keep-Alive" header. The "Keep-Alive" header provides Hypertext Transfer Protocol (HTTP) [I-D.ietf-httpbis-p1-messaging] clients, servers and intermediaries with information about the connection use policies of their peers.

The "timeout" Keep-Alive parameter indicates the time that a connection will be allowed to remain idle before it is closed.

Some HTTP implementations already provide an implementation for this header. Not all of those implementations are interoperable due to significant differences in the header format. This draft defines a single format for the header and ascribes specific semantics to the header parameters.

1.1. Idle Connection Timeouts and Connection Reuse

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 a usable connection: establishing a new connection adds significantly to the latency of the request.

Connection resources can be more efficiently used when an idle connection timeout is known. A client that only periodically sends request can learn about the possibility of a connection timeout and can act to create a new connection for requests or send requests that keep the connection from timing out. Alternatively, a client that knows that more requests on a connection are unlikely within the discovered timeout interval can close the connection immediately after a poll, releasing resources.

1.2. Terminology

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 [RFC2119] and indicate requirement levels for compliant implementations.

2. Keep-Alive Header

The "Keep-Alive" header is a hop-by-hop header that provides information about a persistent connection. Both client and server are able to provide information independently.

```
Keep-Alive           = "Keep-Alive" ":" 1#keep-alive-info
keep-alive-info      =   "timeout" "=" delta-seconds
                       / keep-alive-extension
keep-alive-extension = token [ "=" ( token / quoted-string ) ]
```

This header is sent by either host participating in a persistent connection. The values might be set based on policy implemented by servers, clients and intermediaries. Values might also be set based on knowledge that a host has about lower layer intermediaries in the path of the request, such as a TCP middlebox. Such middleboxes, in particular network address translators (NATs), frequently discard mappings for idle connections, causing the connection to fail after a certain duration of inactivity.

The value of Keep-Alive parameters can change on each request or response sent on a connection. Absence of the header or any parameter implies that any previously provided value still applies.

As a hop-by-hop header, this header only applies to a single transport-level connection. If a Keep-Alive header is added to a request or response, the Connection header MUST include the tag "Keep-Alive". This ensures that compliant intermediaries that do not recognize this header remove it before forwarding a request.

2.1. 'timeout' Parameter

A host sets the value of the "timeout" parameter to the time that the host will allow an idle connection to remain open before it is closed. A connection is idle if no data is sent or received by a host.

The value of the "timeout" parameter is a single integer in seconds.

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.

Each peer, client or server, has a different view of the time that a connection becomes idle. Packet transmission at one peer necessarily occurs before receipt, meaning that the sending peer perceives the connection as being idle earlier to the receiving peer. Similarly, the buffering or retransmission of data by lower layers of the stack,

which is unlikely to be visible to the HTTP implementation, compounds this effect. Clients are advised to make allowances for delays in determining whether to reuse an idle connection.

2.2. Other Header Parameters

The Keep-Alive header can be extended by adding any number of keep-alive-extension values to the header. Any extension that is not understood MUST be ignored.

The HTTP Keep-Alive Information Registry defines the namespace for Keep-Alive extensions. Section 7.2 describes this registry.

2.2.1. 'max' Parameter

The "max" parameter has been used to indicate the maximum number of requests that would be made on the connection. This parameter is deprecated. Any limit on requests can be enforced by sending "Connection: close" and closing the connection.

3. Existing Intermediaries

The exact impact of an intermediary on an HTTP request with a Keep-Alive header depends on the type of intermediary.

An intermediary that is compliant with HTTP/1.1, but does not implement Keep-Alive, 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]).

A non-compliant "transparent" intermediary could pass this header on to the next hop. This results in errors of the sort that are described in Section A.1.2 of [I-D.ietf-httpbis-p1-messaging]. Clients that send this header to HTTP/1.0 servers or proxies SHOULD monitor for "hung" connections and avoid sending the header if a connection appears to hang.

A network address translation (NAT) device or other middlebox might cause a connection to become unavailable prior to the advertised timeout.

A client or intermediary can revise or remove the Keep-Alive header for subsequent requests to the same resource or origin server if it detects non-compliant intermediaries or middleboxes that have shorter timeout periods.

4. Upgraded HTTP Connections

A connection timeout can apply to a connection that is subsequently upgraded to another protocol [RFC2817], such as the websocket protocol [RFC6455].

The idle connection timeout applies to the upgraded connection, unless the upgraded protocol provides another method for indicating idle timeouts.

A server, client or intermediary might apply different policies to an upgraded protocol.

Upgraded protocols might establish an end-to-end connection. As a hop-by-hop header, the values in the "Keep-Alive" on each hop apply to every hop equally. For "timeout", this means that the lowest value from any hop applies to the connection.

Intermediaries that support this header SHOULD determine the impact of a header parameter on dependent hops and reflect that in the values they set. For "timeout", this means that the lowest value from the values seen and the local value is provided in outgoing messages.

5. Examples

The example in Figure 1 shows how a Keep-Alive 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 prepared to retain the connection for up to 3600 seconds (1 hour). On the link between proxy and server, the proxy requests a timeout of 1200 seconds and the server indicates a lower limit of 300 seconds.

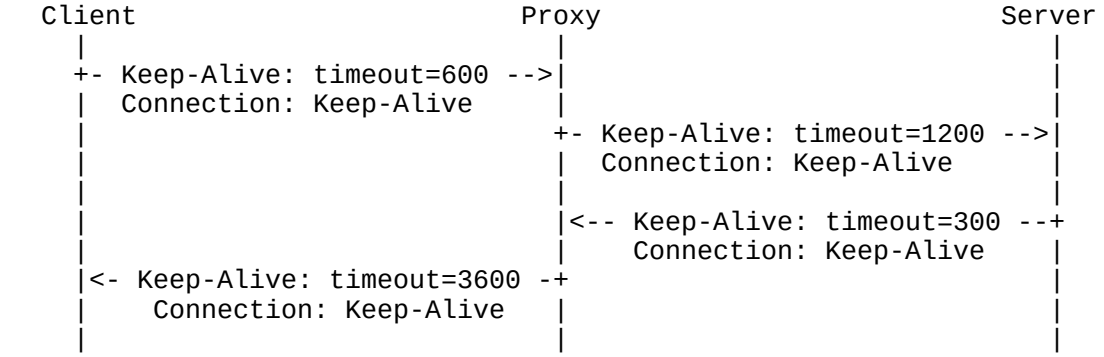

Figure 1: Independent HTTP Hops

As this example shows, the timeout policies maintained by the proxy are different for each connection. Each connection hop is independent.

The example in Figure 2 shows the headers included in an upgrade from HTTP/1.1 to WebSocket [RFC6455]. With a websocket upgrade, the connections on each hop cannot have independent lifecycles on either side of an intermediary. After the upgrade, timeout policies cannot be independent for each hop. The proxy adjusts the timeout value to reflect the lower of the values set by client and the proxy policies so that the server is aware of the connection characteristics; similarly, the value from the server is provided to the client.

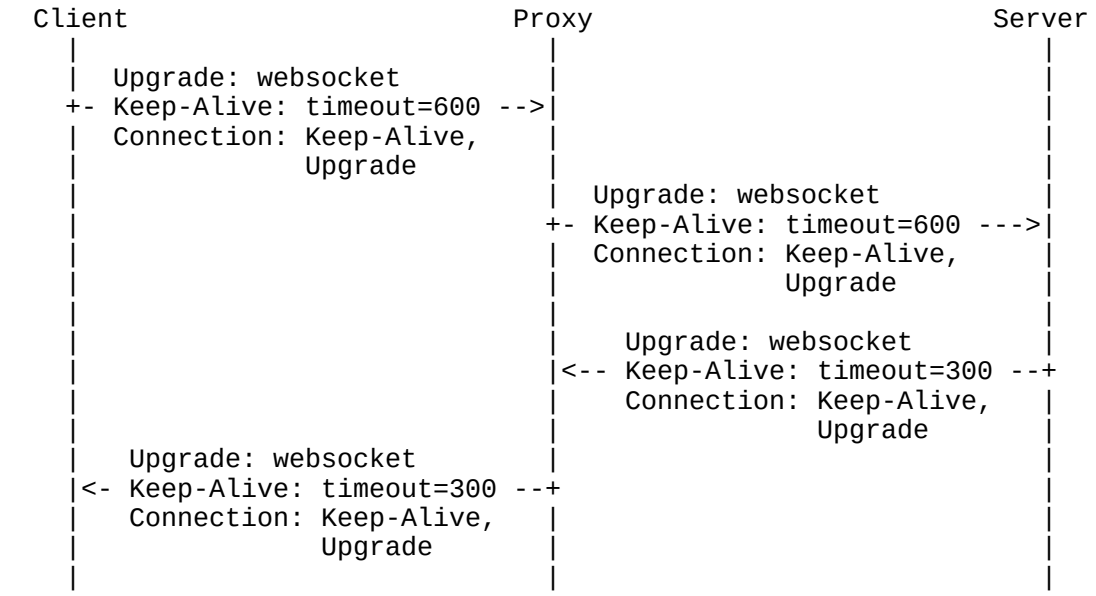

Figure 2: Interdependent Connections with Upgrade

6. Security Considerations

Establishing a persistent connection requires a commitment of resources at a host. The Keep-Alive header are used to express host policy that could alter the way that a host allocates connection resources. Since these policies can be enacted without this feedback, these indications have little effect on security other than exposing specifics of policy.

A host can close a non-idle connection sooner than the indicated time if necessary or as dictated by local policy (see Section 7.1.4 of

[I-D.ietf-httpbis-p1-messaging]).

7. IANA Considerations

[[Note to IANA/RFC Editor: Please replace instance of RFCXXXX with the number of the published RFC and remove this note.]]

7.1. Registration for Keep-Alive Header

This document registers the HTTP "Keep-Alive" header in the "Permanent Message Header Fields" registry established by [RFC3864]

Header field: Keep-Alive

Applicable protocol: HTTP

Status: standard

Author/change controller: Internet Engineering Task Force, IETF (iesg@ietf.org)

Specification document(s): RFCXXXX (this document)

7.2. Registry for Keep-Alive Information

This document establishes a registry for Keep-Alive Information.

Registrations are subject to Specification Required [RFC5226]. The designated expert is advised to review registrations and work with the submitter to ensure that:

- the registration name conforms to the HTTP "token" grammar
- a stable specification exists that is sufficient for interoperable implementation
- the registration does not duplicate an existing entry

The registry includes the following initial values:

timeout See Section 2.1 of this document.

max Deprecated. See Section 2.2.1 of this document.

8. Acknowledgments

Jamie Lokier provided valuable contributions of experience, insight and text suggestions to this document. Roy Fielding provided information on existing implementations of the poorly documented header. Also provided useful feedback: Patrick McManus, Dave Thaler, Konstantinos Pentikousis.

9. Change Log

Since -01:

- Deprecated 'max'
- Corrected badly misleading examples
- Loosened the registry policy from IETF Review

Since -00:

- Removed Request-Timeout in favor of the wait parameter of the Prefer header.
- Connection-Timeout has now been replaced with the zombie spawn of Keep-Alive. This means that it picks up the 'max' parameter as baggage. Open question: should 'max' be deprecated?

Since draft-loreto-http-timeout:

- Changed Timeout to Request-Timeout to avoid a conflict with an existing header definition.
- Added note about the application of Connection-Timeout to upgraded protocols.

10. References

10.1. Normative References

[I-D.ietf-httpbis-p1-messaging]
Fielding, R., Lafon, Y., and J. Reschke, "HTTP/1.1, part 1: Message Routing and Syntax"", draft-ietf-httpbis-p1-messaging-20 (work in progress), July 2012.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

10.2. Informative References

[RFC2068] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

[RFC2817] Khare, R. and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

Authors' Addresses

Martin Thomson
Microsoft
3210 Porter Drive
Palo Alto, CA 94304
US

Phone: +1 650-353-1925
Email: martin.thomson@skype.net

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: salvatore.loreto@ericsson.com

Greg Wilkins
Intalio
644 Emerson Street, Suite 200
Palo Alto 94301
USA

Email: gregw@intalio.com