QUIC
Internet-Draft
Intended status: Informational
Expires: 17 May 2026

L. Pardue
Cloudflare

13 November 2025

QUIC Idle Timeout Update
draft-pardue-quic-idle-timeout-update-00

Abstract

QUIC supports an idle timeout for connections, which can be negotiated once during the connection handshake. This document defines QUIC extension frames that permit either endpoint to initiate an update to the idle timeout value.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://LPardue.github.io/draft-pardue-quic-idle-timeout-update/draft-pardue-quic-idle-timeout-update.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-pardue-quic-idle-timeout-update/.

Discussion of this document takes place on the QUIC Working Group mailing list (mailto:quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/quic/. Subscribe at https://www.ietf.org/mailman/listinfo/quic/.

Source for this draft and an issue tracker can be found at https://github.com/LPardue/draft-pardue-quic-idle-timeout-update.

Status of This Memo

This Internet-Draft will expire on 17 May 2026.

Table of Contents

1. Introduction

QUIC supports an idle timeout for connections, which can be negotiated once during the connection handshake via the max_idle_timeout transport parameter (Section 18.2 of [RFC9000]). If an idle timeout value is negotiated, connections apply the guidance in Section 10.1 of [RFC9000].

The requirement to negotiate idle timeout during the handshake creates some operational friction including:

* Requiring all application protocols that could be negotiated in a single flight to abide by the same value.

* Requiring all applications that use an application protocol to abide by the same value.

* Limiting the ability for endpoints to change the idle timeout value (including disabling) during the lifetime of a connection. For example, adjusting idle timeouts to meet application behavior, or after initiating a graceful shutdown.

This document defines QUIC extension frames that can be used by either endpoint to update the value of the idle timeout: IDLE_TIMEOUT_UPDATE_REQUEST and IDLE_TIMEOUT_UPDATE_RESPONSE.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses terms, definitions, and notational conventions described in Section 1.2 and Section 1.3 of [RFC9000].

3. Negotiating Extension Use

The idle_timeout_update transport parameter (value TBD-01) is defined for QUIC version 1 [RFC9000]. This transport parameter can be sent by both client and server. The transport parameter is sent with an empty value; an endpoint that understands this transport parameter MUST treat receipt of a non-empty value of the transport parameter as a connection error of type TRANSPORT_PARAMETER_ERROR.

The extension frames defined in this document, MUST be sent only when both endpoints advertise the idle_timeout_update transport parameter. If only one endpoint advertises idle_timeout_update, the receipt of a either an IDLE_TIMEOUT_UPDATE_REQUEST or IDLE_TIMEOUT_UPDATE_RESPONSE frame is a connection error of type FRAME_ENCODING_ERROR.

4. IDLE_TIMEOUT_UPDATE_REQUEST Frame

The IDLE_TIMEOUT_UPDATE_REQUEST frame (value TBD-02) is used to request an update to the value of the idle timeout on the connection over which it is sent. It can be sent by both client and server. It is structured as follows:

```
IDLE_TIMEOUT_UPDATE_REQUEST Frame {
  Type (i) = TBD-02,
  Sequence Number (i),
  Idle Timeout (i),
}
```

IDLE_TIMEOUT_UPDATE_REQUEST frames contain the following fields:

Sequence Number: The sequence number assigned to the update request, encoded as a variable-length integer; see Section 7.

Idle Timeout: The requested idle timeout value in milliseconds, encoded as a variable-length integer. A value of 0 is a request to disable the idle timeout.

5. IDLE_TIMEOUT_UPDATE_ACCEPT Frame

The IDLE_TIMEOUT_UPDATE_ACCEPT frame (value TBD-03) is used to accept an idle timeout update request. It can be sent by clients or servers.

It is structured as follows:

```
IDLE_TIMEOUT_UPDATE_ACCEPT Frame {
  Type (i) = TBD-03,
  Sequence Number (i),
}
```

IDLE_TIMEOUT_UPDATE_ACCEPT frames contain the following fields:

Sequence Number: The sequence number that the request being responded to, encoded as a variable-length integer; see Section 7.

6. IDLE_TIMEOUT_UPDATE_REJECT Frame

The IDLE_TIMEOUT_UPDATE_REJECT frame (value TBD-04) is used to reject an idle timeout update request. It can be sent by clients or servers.

It is structured as follows:

```
IDLE_TIMEOUT_UPDATE_REJECT Frame {
  Type (i) = TBD-04,
  Sequence Number (i),
}
```

IDLE_TIMEOUT_UPDATE_REJECT frames contain the following fields:

Sequence Number: The sequence number that the request being responded to, encoded as a variable-length integer; see Section 7.

7. Behavior and Usage

At any point after the handshake, either client or server may request an update to the connection idle timeout by sending an IDLE_TIMEOUT_UPDATE_REQUEST frame containing a sequence number and a requested value. Clients MUST use even-numbered sequence numbers, starting at 0. Servers MUST use odd-numbered sequence numbers, starting at 1. This avoids race conditions with sequence number allocation and prevents an endpoint accepting its own request. When receiving an IDLE_TIMEOUT_UPDATE_REQUEST frame, clients MUST treat an even-numbered sequence number, and servers MUST tread an odd-numbered sequence number, as a connection error of type FRAME_ENCODING_ERROR.

Each update request MUST increase the sequence number by 1.

Recipients of an IDLE_TIMEOUT_UPDATE_REQUEST frame SHOULD respond with either an IDLE_TIMEOUT_UPDATE_ACCEPT or IDLE_TIMEOUT_UPDATE_REJECT frame containing the sequence number of the request that is being responded to. When receiving an IDLE_TIMEOUT_UPDATE_ACCEPT or IDLE_TIMEOUT_UPDATE_REJECT frame, clients MUST treat an odd-numbered sequence number, and servers MUST tread an even-numbered sequence number, as a connection error of type FRAME_ENCODING_ERROR.

Endpoints are not required to respond to every IDLE_TIMEOUT_UPDATE_REQUEST. Responding only to the request with the largest received sequence number can ensure the peer’s most-recent request is accepted, and avoid needing to send explicit rejections for stale values.

Each update request SHOULD contain an idle timeout value different from the immediately previous request. Recipients of IDLE_TIMEOUT_UPDATE_REQUEST frames where the value does not change MAY treat this as a connection error of type FRAME_ENCODING_ERROR.

Accepting an idle timeout update request changes the connection idle timeout value. The update requestor applies the new timeout value on receipt of the IDLE_TIMEOUT_UPDATE_ACCEPT frame, the update responder applies the new timeout value on receipt of acknowledgement of the IDLE_TIMEOUT_UPDATE_ACCEPT frame.

Requesting a timeout value of 0 is valid. If it is accepted, then the idle timeout is disabled.

Rejecting an idle timeout update request leaves the current connection idle timeout value unchanged.

All behavior related to idle connections as described in Section 10.1 of [RFC9000] applies.

8. Security Considerations

The considerations in [RFC9000] apply. Notably, the frame processing requirements in this draft should be aware for the potential for abuse described in Section 21.9 of [RFC9000].

A malicious peer could manipulate congestion control to prevent the sending of update responses and issue a large number of update requests. Endpoints are advised to avoid excessive queuing of pending update responses. Responding only to the largest received sequence number is one strategy to avoid queuing.

9. IANA Considerations

This document provisionally registers a new value in the "QUIC Transport Parameters" registry maintained at https://www.iana.org/assignments/quic (https://www.iana.org/assignments/quic).

Value: TBD-01 (0x0c02ce490eceab89)

Parameter Name: idle_timeout_update

Status: provisional

Specification: This document

Note: Deterministically generated via https://martinthomson.github.io/quic-pick/#seed=draft-pardue-quic-idle-timeout-00-tp;field=frame;codepoint=0x0c02ce490eceab89;size=8 (https://martinthomson.github.io/quic-pick/#seed=draft-pardue-quic-idle-timeout-00-tp;field=frame;codepoint=0x0c02ce490eceab89;size=8)

This document provisionally registers new values QUIC Frame Types" registry maintained at https://www.iana.org/assignments/quic (https://www.iana.org/assignments/quic).

Value: TBD-02 (0x00935f270e717f68)

Frame Name: IDLE_TIMEOUT_UPDATE_REQUEST

Status: provisional

Specification: This document

Note: Deterministically generated via https://martinthomson.github.io/quic-pick/#seed=draft-pardue-quic-idle-timeout-00-req;field=frame;codepoint=0x00935f270e717f68;size=8 (https://martinthomson.github.io/quic-pick/#seed=draft-pardue-quic-idle-timeout-00-req;field=frame;codepoint=0x00935f270e717f68;size=8)

Value: TBD-03 (0x07f531ea3d7b9654)

Frame Name: IDLE_TIMEOUT_UPDATE_ACCEPT

Status: provisional

Specification: This document

Note: Deterministically generated via https://martinthomson.github.io/quic-pick/#seed=draft-pardue-quic-idle-timeout-00-resp;field=frame;codepoint=0x07f531ea3d7b9654;size=8 (https://martinthomson.github.io/quic-pick/#seed=draft-pardue-quic-idle-timeout-00-resp;field=frame;codepoint=0x07f531ea3d7b9654;size=8)

Value: TBD-04 (0x07f531ea3d7b9655)

Frame Name: IDLE_TIMEOUT_UPDATE_REJECT

Status: provisional

Specification: This document

Note: Deterministically generated via TBD-03 value +1

10. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

Acknowledgments

TODO acknowledge.

Author's Address

Lucas Pardue
Cloudflare
Email: lucas@lucaspardue.com