

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 25 February 2021

T. Pauly  
E. Kinnear  
Apple Inc.  
D. Schinazi  
Google LLC  
24 August 2020

An Unreliable Datagram Extension to QUIC  
draft-ietf-quic-datagram-01

## Abstract

This document defines an extension to the QUIC transport protocol to add support for sending and receiving unreliable datagrams over a QUIC connection.

Discussion of this work is encouraged to happen on the QUIC IETF mailing list [quic@ietf.org](mailto:quic@ietf.org) (<mailto:quic@ietf.org>) or on the GitHub repository which contains the draft: <https://github.com/quicwg/datagram> (<https://github.com/quicwg/datagram>).

## 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 <https://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 25 February 2021.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights

Internet-Draft

QUIC Datagrams

August 2020

and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">2</a>
<a href="#">1.1.</a>	Specification of Requirements . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Motivation . . . . .	<a href="#">3</a>
<a href="#">3.</a>	Transport Parameter . . . . .	<a href="#">4</a>
<a href="#">4.</a>	Datagram Frame Type . . . . .	<a href="#">5</a>
<a href="#">5.</a>	Behavior and Usage . . . . .	<a href="#">5</a>
<a href="#">5.1.</a>	Acknowledgement Handling . . . . .	<a href="#">6</a>
<a href="#">5.2.</a>	Flow Control . . . . .	<a href="#">6</a>
<a href="#">5.3.</a>	Congestion Control . . . . .	<a href="#">7</a>
<a href="#">6.</a>	Security Considerations . . . . .	<a href="#">7</a>
<a href="#">7.</a>	IANA Considerations . . . . .	<a href="#">7</a>
<a href="#">8.</a>	Acknowledgments . . . . .	<a href="#">8</a>
<a href="#">9.</a>	References . . . . .	<a href="#">8</a>
<a href="#">9.1.</a>	Normative References . . . . .	<a href="#">8</a>
<a href="#">9.2.</a>	Informative References . . . . .	<a href="#">8</a>
	Authors' Addresses . . . . .	<a href="#">9</a>

## [1.](#) Introduction

The QUIC Transport Protocol [[I-D.ietf-quic-transport](#)] provides a secure, multiplexed connection for transmitting reliable streams of application data. Reliability within QUIC is performed on a per-stream basis, so some frame types are not eligible for retransmission.

Some applications, particularly those that need to transmit real-time data, prefer to transmit data unreliably. These applications can build directly upon UDP [[RFC0768](#)] as a transport, and can add security with DTLS [[RFC6347](#)]. Extending QUIC to support transmitting unreliable application data would provide another option for secure datagrams, with the added benefit of sharing a cryptographic and authentication context used for reliable streams.

This document defines two new DATAGRAM QUIC frame types, which carry application data without requiring retransmissions.

Discussion of this work is encouraged to happen on the QUIC IETF mailing list [quic@ietf.org](mailto:quic@ietf.org) (<mailto:quic@ietf.org>) or on the GitHub repository which contains the draft: <https://github.com/quicwg/datagram> (<https://github.com/quicwg/datagram>).

### [1.1.](#) Specification of Requirements

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.

## [2.](#) Motivation

Transmitting unreliable data over QUIC provides benefits over existing solutions:

- \* Applications that open both a reliable TLS stream and an unreliable DTLS flow to the same peer can benefit by sharing a single handshake and authentication context between a reliable QUIC stream and flow of unreliable QUIC datagrams. This can reduce the latency required for handshakes.
- \* QUIC uses a more nuanced loss recovery mechanism than the DTLS handshake, which has a basic packet loss retransmission timer. This may allow loss recovery to occur more quickly for QUIC data.
- \* QUIC datagrams, while unreliable, can support acknowledgements, allowing applications to be aware of whether a datagram was successfully received.
- \* QUIC datagrams are subject to QUIC congestion control, allowing applications to avoid implementing their own.

These reductions in connection latency, and application insight into the delivery of datagrams, can be useful for optimizing audio/video streaming applications, gaming applications, and other real-time network applications.

Unreliable QUIC datagrams can also be used to implement an IP packet

tunnel over QUIC, such as for a Virtual Private Network (VPN). Internet-layer tunneling protocols generally require a reliable and authenticated handshake, followed by unreliable secure transmission of IP packets. This can, for example, require a TLS connection for the control data, and DTLS for tunneling IP packets. A single QUIC connection could support both parts with the use of unreliable datagrams.

### [3.](#) Transport Parameter

Support for receiving the DATAGRAM frame types is advertised by means of a QUIC Transport Parameter (name=max\_datagram\_frame\_size, value=0x0020). The max\_datagram\_frame\_size transport parameter is an integer value (represented as a variable-length integer) that represents the maximum size of a DATAGRAM frame (including the frame type, length, and payload) the endpoint is willing to receive, in bytes. An endpoint that includes this parameter supports the DATAGRAM frame types and is willing to receive such frames on this connection. Endpoints MUST NOT send DATAGRAM frames until they have received the max\_datagram\_frame\_size transport parameter. Endpoints MUST NOT send DATAGRAM frames of size strictly larger than the value of max\_datagram\_frame\_size the endpoint has received from its peer. An endpoint that receives a DATAGRAM frame when it has not sent the max\_datagram\_frame\_size transport parameter MUST terminate the connection with error `PROTOCOL_VIOLATION`. An endpoint that receives a DATAGRAM frame that is strictly larger than the value it sent in its max\_datagram\_frame\_size transport parameter MUST terminate the connection with error `PROTOCOL_VIOLATION`. Endpoints that wish to use DATAGRAM frames need to ensure they send a max\_datagram\_frame\_size value sufficient to allow their peer to use them. It is RECOMMENDED to send the value 65536 in the max\_datagram\_frame\_size transport parameter as that indicates to the peer that this endpoint will accept any DATAGRAM frame that fits inside a QUIC packet.

The max\_datagram\_frame\_size transport parameter is a unidirectional limit and indication of support of DATAGRAM frames. Application protocols that use DATAGRAM frames MAY choose to only negotiate and



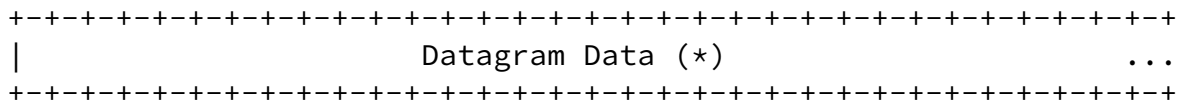


Figure 1: DATAGRAM Frame Format

DATAGRAM frames contain the following fields:

**Length:** A variable-length integer specifying the length of the datagram in bytes. This field is present only when the LEN bit is set. If the LEN bit is not set, the datagram data extends to the end of the QUIC packet. Note that empty (i.e., zero-length) datagrams are allowed.

**Datagram Data:** The bytes of the datagram to be delivered.

## 5. Behavior and Usage

When an application sends an unreliable datagram over a QUIC connection, QUIC will generate a new DATAGRAM frame and send it in the first available packet. This frame SHOULD be sent as soon as possible, and MAY be coalesced with other frames.

When a QUIC endpoint receives a valid DATAGRAM frame, it SHOULD deliver the data to the application immediately, as long as it is able to process the frame and can store the contents in memory.

DATAGRAM frames MUST be protected with either 0-RTT or 1-RTT keys.

Application protocols using datagrams are responsible for defining the semantics of the Datagram Data field, and how it is parsed. If the application protocol supports the coexistence of multiple entities using datagrams inside a single QUIC connection, it may need a mechanism to allow demultiplexing between them. For example, using datagrams with HTTP/3 involves prepending a flow identifier to all datagrams, see [[I-D.schinazi-quic-h3-datagram](#)].

Note that while the `max_datagram_frame_size` transport parameter places a limit on the maximum size of DATAGRAM frames, that limit can be further reduced by the `max_packet_size` transport parameter, and by the Maximum Transmission Unit (MTU) of the path between endpoints. DATAGRAM frames cannot be fragmented, therefore application protocols

need to handle cases where the maximum datagram size is limited by other factors.

### [5.1.](#) Acknowledgement Handling

Although DATAGRAM frames are not retransmitted upon loss detection, they are ack-eliciting ([\[I-D.ietf-quic-recovery\]](#)). Receivers SHOULD support delaying ACK frames (within the limits specified by `max_ack_delay`) in response to receiving packets that only contain DATAGRAM frames, since the timing of these acknowledgements is not used for loss recovery.

If a sender detects that a packet containing a specific DATAGRAM frame might have been lost, the implementation MAY notify the application that it believes the datagram was lost. Similarly, if a packet containing a DATAGRAM frame is acknowledged, the implementation MAY notify the application that the datagram was successfully transmitted and received. Note that, due to reordering, a DATAGRAM frame that was thought to be lost could at a later point be received and acknowledged.

### [5.2.](#) Flow Control

DATAGRAM frames do not provide any explicit flow control signaling, and do not contribute to any per-flow or connection-wide data limit.

The risk associated with not providing flow control for DATAGRAM frames is that a receiver may not be able to commit the necessary resources to process the frames. For example, it may not be able to store the frame contents in memory. However, since DATAGRAM frames are inherently unreliable, they MAY be dropped by the receiver if the receiver cannot process them.

### [5.3.](#) Congestion Control

DATAGRAM frames employ the QUIC connection's congestion controller. As a result, a connection may be unable to send a DATAGRAM frame generated by the application until the congestion controller allows it [\[I-D.ietf-quic-recovery\]](#). The sender implementation MUST either delay sending the frame until the controller allows it or drop the frame without sending it (at which point it MAY notify the

application).

Implementations can optionally support allowing the application to specify a sending expiration time, beyond which a congestion-controlled DATAGRAM frame ought to be dropped without transmission.

## [6.](#) Security Considerations

The DATAGRAM frame shares the same security properties as the rest of the data transmitted within a QUIC connection. All application data transmitted with the DATAGRAM frame, like the STREAM frame, MUST be protected either by 0-RTT or 1-RTT keys.

## [7.](#) IANA Considerations

This document registers a new value in the QUIC Transport Parameter Registry:

Value: 0x0020 (if this document is approved)

Parameter Name: max\_datagram\_frame\_size

Specification: Indicates that the connection should enable support for unreliable DATAGRAM frames. An endpoint that advertises this transport parameter can receive datagrams frames from the other endpoint, up to and including the length in bytes provided in the transport parameter.

This document also registers a new value in the QUIC Frame Type registry:

Value: 0x30 and 0x31 (if this document is approved)

Frame Name: DATAGRAM

Specification: Unreliable application data

## [8.](#) Acknowledgments



Thanks to Ian Swett, who inspired this proposal.

## 9. References

### 9.1. Normative References

[I-D.ietf-quick-recovery]

Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", Work in Progress, Internet-Draft, [draft-ietf-quick-recovery-29](http://www.ietf.org/internet-drafts/draft-ietf-quick-recovery-29), 9 June 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-quick-recovery-29.txt>>.

[I-D.ietf-quick-transport]

Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, [draft-ietf-quick-transport-29](http://www.ietf.org/internet-drafts/draft-ietf-quick-transport-29), 9 June 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-quick-transport-29.txt>>.

### 9.2. Informative References

[I-D.schinazi-quick-h3-datagram]

Schinazi, D., "Using QUIC Datagrams with HTTP/3", Work in Progress, Internet-Draft, [draft-schinazi-quick-h3-datagram-04](http://www.ietf.org/internet-drafts/draft-schinazi-quick-h3-datagram-04), 16 April 2020, <<http://www.ietf.org/internet-drafts/draft-schinazi-quick-h3-datagram-04.txt>>.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, [RFC 768](https://www.rfc-editor.org/info/rfc768), DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.

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

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", [RFC 6347](https://www.rfc-editor.org/info/rfc6347), DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.

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

Authors' Addresses

Tommy Pauly  
Apple Inc.  
One Apple Park Way  
Cupertino, California 95014,  
United States of America

Email: [tpauly@apple.com](mailto:tpauly@apple.com)

Eric Kinnear  
Apple Inc.  
One Apple Park Way  
Cupertino, California 95014,  
United States of America

Email: [ekinnear@apple.com](mailto:ekinnear@apple.com)

David Schinazi  
Google LLC  
1600 Amphitheatre Parkway  
Mountain View, California 94043,  
United States of America

Email: [dschinazi.ietf@gmail.com](mailto:dschinazi.ietf@gmail.com)

