Workgroup: Network Working Group
Internet-Draft:
draft-schinazi-masque-h3-datagram-04
Published: 5 January 2021
Intended Status: Standards Track
Expires: 9 July 2021
Authors: D. Schinazi L. Pardue
Google LLC Cloudflare

Using QUIC Datagrams with HTTP/3

Abstract

The QUIC DATAGRAM extension provides application protocols running over QUIC with a mechanism to send unreliable data while leveraging the security and congestion-control properties of QUIC. However, QUIC DATAGRAM frames do not provide a means to demultiplex application contexts. This document defines how to use QUIC DATAGRAM frames when the application protocol running over QUIC is HTTP/3 by adding an identifier at the start of the frame payload. This allows HTTP messages to convey related information using unreliable DATAGRAM frames, ensuring those frames are properly associated with an HTTP message.

Discussion of this work is encouraged to happen on the MASQUE IETF mailing list (masque@ietf.org) or on the GitHub repository which contains the draft: https://github.com/DavidSchinazi/draft-h3-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 9 July 2021.

Copyright Notice

Copyright (c) 2021 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 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

1. Introduction

The QUIC DATAGRAM extension [DGRAM] provides application protocols running over QUIC [QUIC] with a mechanism to send unreliable data while leveraging the security and congestion-control properties of QUIC. However, QUIC DATAGRAM frames do not provide a means to demultiplex application contexts. This document defines how to use QUIC DATAGRAM frames when the application protocol running over QUIC is HTTP/3 [H3] by adding an identifier at the start of the frame payload. This allows HTTP messages to convey related information using unreliable DATAGRAM frames, ensuring those frames are properly associated with an HTTP message.

This design mimics the use of Stream Types in HTTP/3, which provide a demultiplexing identifier at the start of each unidirectional stream.

Discussion of this work is encouraged to happen on the MASQUE IETF mailing list (masque@ietf.org) or on the GitHub repository which contains the draft: https://github.com/DavidSchinazi/draft-h3-datagram.

1.1. 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.

2. Flow Identifiers

Flow identifiers represent bidirectional flows of datagrams within a single QUIC connection. These are conceptually similar to streams in the sense that they allow multiplexing of application data. Flows lack any of the ordering or reliability guarantees of streams.

Beyond this, a sender SHOULD ensure that DATAGRAM frames within a single flow are transmitted in order relative to one another. If multiple DATAGRAM frames can be packed into a single QUIC packet, the sender SHOULD group them by flow identifier to promote fate-sharing within a specific flow and improve the ability to process batches of datagram messages efficiently on the receiver.

3. Flow Identifier Allocation

Implementations of HTTP/3 that support the DATAGRAM extension MUST provide a flow identifier allocation service. That service will allow applications co-located with HTTP/3 to request a unique flow identifier that they can subsequently use for their own purposes. The HTTP/3 implementation will then parse the flow identifier of incoming DATAGRAM frames and use it to deliver the frame to the appropriate application.

Even-numbered flow identifiers are client-initiated, while odd-numbered flow identifiers are server-initiated. This means that an HTTP/3 client implementation of the flow identifier allocation service MUST only provide even-numbered identifiers, while a server implementation MUST only provide odd-numbered identifiers. Note that, once allocated, any flow identifier can be used by both client and server - only allocation carries separate namespaces to avoid requiring synchronization.

The flow allocation service SHOULD also provide a mechanism for applications to indicate they have completed their usage of a flow identifier and will no longer be using that flow identifier, this process is called "retiring" a flow identifier. Applications MUST NOT retire a flow identifier until after they have received confirmation that the peer has also stopped using that flow identifier. The flow identifier allocation service MAY reuse previously retired flow identifiers once they have ascertained that there are no packets with DATAGRAM frames using that flow identifier still in flight. Reusing flow identifiers can improve performance by transmitting the flow identifier using a shorter variable-length integer encoding.

4. HTTP/3 DATAGRAM Frame Format

When used with HTTP/3, the Datagram Data field of QUIC DATAGRAM frames uses the following format (using the notation from the "Notational Conventions" section of [QUIC]):

```
HTTP/3 DATAGRAM Frame {
  Flow Identifier (i),
  HTTP/3 Datagram Payload (..),
}
```

Figure 1: HTTP/3 DATAGRAM Frame Format

Flow Identifier: A variable-length integer indicating the Flow Identifier of the datagram (see Section 2).

HTTP/3 Datagram Payload: The payload of the datagram, whose semantics are defined by individual applications. Note that this field can be empty.

Endpoints MUST treat receipt of a DATAGRAM frame whose payload is too short to parse the flow identifier as an HTTP/3 connection error of type H3_GENERAL_PROTOCOL_ERROR.

5. The H3_DATAGRAM HTTP/3 SETTINGS Parameter

Implementations of HTTP/3 that support this mechanism can indicate that to their peer by sending the H3_DATAGRAM SETTINGS parameter with a value of 1. The value of the H3_DATAGRAM SETTINGS parameter MUST be either 0 or 1. A value of 0 indicates that this mechanism is not supported. An endpoint that receives the H3_DATAGRAM SETTINGS parameter with a value that is neither 0 or 1 MUST terminate the connection with error H3_SETTINGS_ERROR.

An endpoint that sends the H3_DATAGRAM SETTINGS parameter with a value of 1 MUST send the max_datagram_frame_size QUIC Transport

Parameter [DGRAM]. An endpoint that receives the H3_DATAGRAM SETTINGS parameter with a value of 1 on a QUIC connection that did not also receive the max_datagram_frame_size QUIC Transport Parameter MUST terminate the connection with error H3_SETTINGS_ERROR.

When clients use 0-RTT, they MAY store the value of the server's H3_DATAGRAM SETTINGS parameter. Doing so allows the client to use HTTP/3 datagrams in 0-RTT packets. When servers decide to accept 0-RTT data, they MUST send a H3_DATAGRAM SETTINGS parameter greater than or equal to the value they sent to the client in the connection where they sent them the NewSessionTicket message. If a client stores the value of the H3_DATAGRAM SETTINGS parameter with their 0-RTT state, they MUST validate that the new value of the H3_DATAGRAM SETTINGS parameter sent by the server in the handshake is greater than or equal to the stored value; if not, the client MUST terminate the connection with error H3_SETTINGS_ERROR. In all cases, the maximum permitted value of the H3_DATAGRAM SETTINGS parameter is 1.

6. Datagram-Flow-Id Header Field Definition

"Datagram-Flow-Id" is a List Structured Field [STRUCT-FIELD], whose members MUST all be Items of type Integer. Its ABNF is:

```
Datagram-Flow-Id = sf-list
```

The "Datagram-Flow-Id" header field is used to associate one or more datagram flow identifiers with an HTTP message. As a simple example using a single identifier, the definition of an HTTP method could instruct the client to use its flow identifier allocation service to allocate a new flow identifier, and then the client will add the "Datagram-Flow-Id" header field to its request to communicate that value to the server. In this example, the resulting header field could look like:

```
Datagram-Flow-Id = 2
```

List members are flow identifier elements, which can be named or unnamed. One element in the list is allowed to be unnamed, but all but one elements MUST carry a name. The name of an element is encoded in the key of the first parameter of that element (parameters are defined in Section 3.1.2 of [STRUCT-FIELD]). Each name MUST NOT appear more than once in the list. The value of the first parameter of each named element (whose corresponding key conveys the element name) MUST be of type Boolean and equal to true. The value of the first parameter of the unnamed element MUST NOT be of type Boolean. The ordering of the list does not carry any semantics. For example, an HTTP method that wishes to use four

datagram flow identifiers for the lifetime of its request stream could look like this:

```
Datagram-Flow-Id = 42, 44; ecn-ect0, 46; ecn-ect1, 48; ecn-ce
```

In this example, 42 is the unnamed flow identifier, 44 represents the name "ecn-ect0", 46 represents "ecn-ect1", and 48 represents "ecn-ce". Note that, since the list ordering does not carry semantics, this example can be equivalently encoded as:

```
Datagram-Flow-Id = 44; ecn-ect0, 42, 48; ecn-ce, 46; ecn-ect1
```

Even if a sender attempts to communicate the meaning of a flow identifier before it uses it in an HTTP/3 datagram, it is possible that its peer will receive an HTTP/3 datagram with a flow identifier that it does not know as it has not yet received the corresponding "Datagram-Flow-Id" header field. (For example, this could happen if the QUIC STREAM frame that contains the "Datagram-Flow-Id" header field is reordered and arrives afer the DATAGRAM frame.) Endpoints MUST NOT treat that scenario as an error; they MUST either silently discard the datagram or buffer it until they receive the "Datagram-Flow-Id" header field.

Distinct HTTP requests MAY refer to the same flow identifier in their respective "Datagram-Flow-Id" header fields.

Note that integer structured fields can only encode values up to 10^15-1, therefore the maximum possible value of an element of the "Datagram-Flow-Id" header field is lower then the theoretical maximum value of a flow identifier which is 2^62-1 due to the QUIC variable length integer encoding. If the flow identifier allocation service of an endpoint runs out of values lower than 10^15-1, the endpoint MUST fail the flow identifier allocation. An HTTP message that carries a "Datagram-Flow-Id" header field with a flow identifier value above 10^15-1 is malformed (see Section 8.1.2.6 of [H2]).

7. HTTP Intermediaries

HTTP/3 DATAGRAM flow identifiers are specific to a given HTTP/3 connection. However, in some cases, an HTTP request may travel across multiple HTTP connections if there are HTTP intermediaries involved; see Section 2.3 of [RFC7230].

If an intermediary has sent the H3_DATAGRAM SETTINGS parameter with a value of 1 on its client-facing connection, it MUST inspect all HTTP requests from that connection and check for the presence of the "Datagram-Flow-Id" header field. If the HTTP method of the request is not supported by the intermediary, it MUST remove the "Datagram-Flow-Id" header field before forwarding the request. If the

intermediary supports the method, it MUST either remove the header field or adhere to the requirements leveraged by that method on intermediaries.

If an intermediary has sent the H3_DATAGRAM SETTINGS parameter with a value of 1 on its server-facing connection, it MUST inspect all HTTP responses from that connection and check for the presence of the "Datagram-Flow-Id" header field. If the HTTP method of the request is not supported by the intermediary, it MUST remove the "Datagram-Flow-Id" header field before forwarding the response. If the intermediary supports the method, it MUST either remove the header field or adhere to the requirements leveraged by that method on intermediaries.

If an intermediary processes distinct HTTP requests that refer to the same flow identifier in their respective "Datagram-Flow-Id" header fields, it MUST ensure that those requests are routed to the same backend.

8. Security Considerations

This document does not have additional security considerations beyond those defined in [QUIC] and [DGRAM].

9. IANA Considerations

9.1. HTTP SETTINGS Parameter

This document will request IANA to register the following entry in the "HTTP/3 Settings" registry:

Setting Name	Value	Specification	Default
H3_DATAGRAM	0x276	This Document	0

9.2. HTTP Header Field

This document will request IANA to register the "Datagram-Flow-Id" header field in the "Permanent Message Header Field Names" registry maintained at <https://www.iana.org/assignments/message-headers>.

Header Field Name	Protocol	Status	Reference
Datagram-Flow-Id	http	std	This document

9.3. Flow Identifier Parameters

This document will request IANA to create an "HTTP Datagram Flow Identifier Parameters" registry. Registrations in this registry MUST include the following fields:

Key: The key of a parameter that is associated with a datagram flow identifier list member (see Section 6). Keys MUST be valid structured field parameter keys (see Section 3.1.2 of [STRUCT-FIELD]).

Description: A brief description of the parameter semantics, which MAY be a summary if a specification reference is provided.

Is Name: This field MUST be either Yes or No. Yes indicates that this parameter is the name of a named element (see Section 6). No indicates that it is a parameter that is not a name.

Reference: An optional reference to a specification for the parameter. This field MAY be empty.

Registrations follow the "First Come First Served" policy (see Section 4.4 of [IANA-POLICY]) where two registrations MUST NOT have the same Key. This registry is initially empty.

10. Normative References

[DGRAM] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-datagram-01, 24 August 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-datagram-01.txt>.

[H2] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[H3] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-33, 15 December 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-http-33.txt>.

[IANA-POLICY] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[QUIC] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress,

Internet-Draft, draft-ietf-quic-transport-33, 13 December 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-transport-33.txt>.

[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/info/rfc2119>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[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/info/rfc8174>.

[STRUCT-FIELD] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", Work in Progress, Internet-Draft, draft-ietf-httpbis-header-structure-19, 3 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-httpbis-header-structure-19.txt>.

Acknowledgments

The DATAGRAM flow identifier was previously part of the DATAGRAM frame definition itself, the author would like to acknowledge the authors of that document and the members of the IETF MASQUE working group for their suggestions. Additionally, the author would like to thank Martin Thomson for suggesting the use of an HTTP/3 SETTINGS parameter.

Authors' Addresses

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

Email: dschinazi.ietf@gmail.com

Lucas Pardue
Cloudflare

Email: lucaspardue.24.7@gmail.com