Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2026

J. Yao
H. Li
M. Zhang
CNNIC
D. Keathley
J. Gould
VeriSign, Inc.
16 March 2026

Extensible Provisioning Protocol (EPP) Transport over QUIC
draft-ietf-regext-epp-quic-04

Abstract

This document describes how an Extensible Provisioning Protocol (EPP) session is mapped onto a QUIC connection. EPP over QUIC (EoQ) leverages the performance and security features of the QUIC protocol as an EPP transport.

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 17 September 2026.

Copyright Notice

Table of Contents

1. Introduction

This document describes how the Extensible Provisioning Protocol (EPP [RFC5730]) is mapped onto the QUIC transport [RFC9000]. QUIC is a network protocol that is based on UDP and incorporates native encryption support using TLS [RFC9001]. Though based on UDP, QUIC provides connection semantics like other stateful protocols. This document discusses how EPP implementations can work with this and other features of QUIC while preserving the core EPP semantics.

Commonly used terms in this document are described below.

EoQ: The acronym used for the EPP over QUIC transport that defines the use of QUIC as an EPP transport following the considerations in Section 2.1 of [RFC5730].

EoQ connection: Is a client-initiated bidirectional QUIC stream established on a QUIC connection using the "eoq/0.1" ALPN. The EoQ connection maps to the client-server connection defined in Section 2.1 of [RFC5730], where the server returns an EPP greeting. A QUIC connection supports many EoQ connections.

EoQ session: Is an authenticated EoQ connection, which occurs after a successful EPP <login>.

EoQ Connection Start Packet: Used by the client to complete the creation of an EoQ connection by signaling the server to create the QUIC stream and return the EPP greeting needed for an EPP connection. Section 6 formally defines the EoQ Connection Start Packet.

2. Conventions Used in This Document

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 [BCP14] when, and only when, they appear in all capitals, as shown here.

3. Session Management

Mapping EPP session management facilities onto the QUIC service is accomplished with a combination of a QUIC connection with the "EoQ" ALPN value and client-initiated, bidirectional QUIC streams. QUIC supports four stream types, but EoQ only supports the client-initiated, bidirectional stream type. An EPP session first requires creation of a QUIC connection between two peers, one that initiates the connection request and one that responds to the connection request. The initiating peer is called the "client", and the responding peer is called the "server". An EPP server MUST listen for QUIC connection requests on a standard UDP port assigned by IANA.

A successfully established QUIC connection is automatically secured by the native TLS support that QUIC provides using the "eoq/1.0" ALPN value.

Once the QUIC connection is established, the EPP client MUST then create a bidirectional QUIC stream by sending the EoQ Connection Start Packet (Section 6). [RFC9000] states that "streams are created by sending data". Once the EPP server accepts the QUIC stream, it

reads the EoQ Connection Start Packet (Section 6) and returns an EPP <greeting> to the client on same QUIC stream. After reading the EPP <greeting> message, the EPP client sends EPP commands and receives EPP responses on the same stream. A QUIC stream corresponds to an EPP connection, which is referred to as a EoQ connection. An authenticated QUIC stream, via a successful EPP <login>, corresponds to an EPP session, which is referred to as a EoQ session.

An EPP session is normally ended by the client issuing an EPP <logout> command. A server receiving an EPP <logout> command MUST end the EPP session and close the QUIC stream. A client MAY end an EoQ session by closing the QUIC stream and the server MUST end the EoQ session by closing the QUIC stream.

EoQ connections are established as described in the QUIC transport specification [RFC9000]. During connection establishment, EoQ support is indicated by selecting the Application-Layer Protocol Negotiation (ALPN) token "EoQ" in the crypto handshake.

A single QUIC connection may allow multiple QUIC streams. This means that a single QUIC connection may support multiple EoQ sessions. A server MAY limit the life span of an established EoQ session. EoQ sessions that are inactive for more than a server-defined period MAY be ended by a server closing the QUIC stream. A server MAY close EoQ sessions that have been open and active for longer than a server-defined limit. Once the last QUIC stream for a QUIC connection is closed, the server MAY end the QUIC connection immediately.

4. Message Exchange

Except for the EPP server greeting, EPP messages are initiated by the EPP client in the form of EPP commands. An EPP server MUST return an EPP response to an EPP command on the same QUIC stream that carried the command. If the QUIC stream is closed after a server receives and successfully processes a command but before the response can be returned to the client, the server MAY attempt to undo the effects of the command to ensure a consistent state between the client and the server. EPP commands are idempotent, so processing a command more than once produces the same net effect on the repository as successfully processing the command once.

An EPP client streams EPP commands to an EPP server on an established QUIC stream. A client MAY establish multiple QUIC streams to support multiple EoQ sessions with each EoQ session mapped to a single QUIC stream. A server SHOULD limit a client to a maximum number of QUIC streams per QUIC connection based on server capabilities and operational load.

EPP describes client-server interaction as a command-response exchange where the client sends one command to the server and the server returns one response to the client.

Each EPP data unit MUST contain a single EPP message. Commands MUST be processed independently.

A server SHOULD impose a limit on the amount of time required for a client to issue a well-formed EPP command to reduce the risk associated with a resource exhaustion attack. A server SHOULD end an EoQ session and close the QUIC stream if a well-formed command is not received within the time limit.

A general state machine for an EPP server is described in Section 2 of [RFC5730]. A general client-server message exchange using QUIC transport is illustrated in Figure 1. It shows the exchange over a single QUIC stream of a QUIC connection. Many QUIC streams may open and close during the life of a QUIC connection.

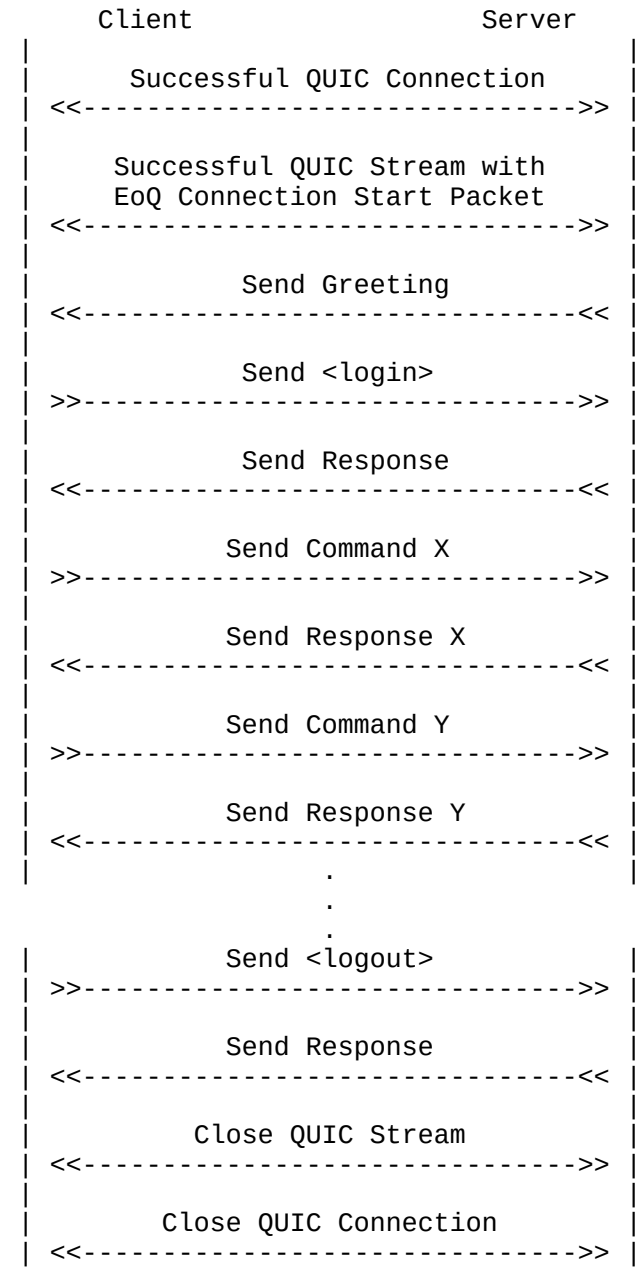
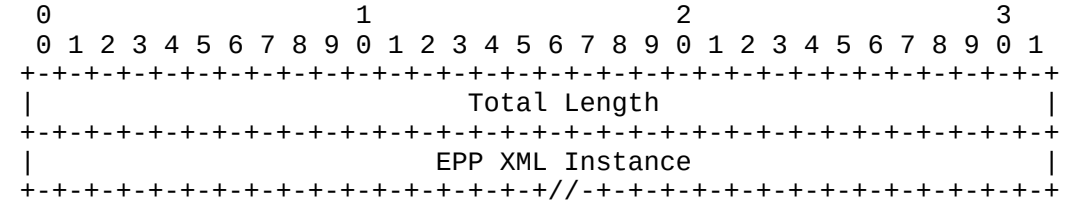


Figure 1: QUIC Client-Server Message Exchange

The EPP server MUST follow the "EPP Server State Machine" procedure described in [RFC5730].

5. Data Unit Format

The EPP data unit contains two fields: a 32-bit header that describes the total length of the data unit, and the EPP XML instance. The length of the EPP XML instance is determined by subtracting four octets from the total length of the data unit. A receiver must successfully read that many octets to retrieve the complete EPP XML instance before processing the EPP message. EPP Data Unit Format (one tick mark represents one bit position):



Total Length (32 bits): The total length of the EPP data unit measured in octets in network (big endian) byte order. The octets contained in this field MUST be included in the total length calculation. EPP XML Instance (variable length): The EPP XML instance carried in the data unit.

6. EoQ Connection Start Packet

The EoQ Connection Start Packet is written by the client after creating a QUIC stream to signal the server to create the QUIC stream. The server accepts the QUIC stream, reads the EoQ Connection Start Packet, and returns the EPP <greeting> to the client on same QUIC stream.

The EoQ Connection Start Packet follows the Data Unit Format (Section 5) with two fields: a 32-bit header that describes the total length of the data unit, and the constant value of "EoQ Connection Start" instead of an "EPP XML Instance". The length of the data unit will be 24 that includes 4 octets for the Total Length and 20 octets for the "EoQ Connection Start" constant value.

7. Transport Considerations

Section 2.1 of [RFC5730] describes considerations to be addressed by protocol transport mappings. This document addresses each of those considerations using a combination of features of the QUIC protocol itself and features of this document.

* Command Order: QUIC guarantees ordered processing of data within each stream. Section 2 of [RFC9000] describes streams in detail.

* Session Mapping: EPP session management utilizes QUIC streams and is described in Section 3

* Stateful Nature: QUIC supports stateful communications between endpoints via connection IDs and long-lived streams within each connection. Sections 2 and 5 of [RFC9000] describe these features, respectively.

* Frame Data Units: EoQ uses the packet framing defined in Section 5.

* Congestion Avoidance: QUIC provides various mechanisms to help achieve congestion avoidance. [RFC9002] describes these mechanisms in detail.

* Reliability: QUIC uses message acknowledgement, packet retransmission, and other features to ensure reliability. Part packetization of [RFC9000] describes these features in detail.

* Pipelining: Pipelining is allowed in EoQ. QUIC streams support sending multiple frames without waiting for responses from the other peer. This does not change the basic single command, single response operating mode of the core EPP protocol.

  Commands MUST be processed independently and in the same order as sent from the client.

  Batch-oriented processing (combining multiple EPP commands in a single data unit) is not permitted. Each EPP data unit MUST contain a single EPP message.

  An EPP x5zz "Connection Management" error response, defined in Section 3 of [RFC5730], of a well-formed EPP client packet results in the server closing the EoQ connection after returning the error response. A malformed EPP client packet results in the server closing the EoQ connection without providing an error response. All subsequent EPP commands sent on the EoQ connection will not be processed.

8. IANA Considerations

8.1. Registration of an EoQ Identification String

This document creates a new registration for the identification of EoQ in the "TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs" registry [RFC7301].

* Protocol: EoQ
* Identification Sequence: 0x45 0x6F 0x51 ("EoQ")
* Reference: This document

8.2. Registration of Port Number

The "Service Name and Transport Protocol Port Number Registry" contains an entry for EPP UDP/700 based on [RFC6335]. However, no known implementations of EPP over UDP exist. The entry will be reassigned to reference this draft.

* Service Name: epp
* Port Number: 700
* Transport Protocol(s): UDP
* Assignee: IESG
* Contact: IETF Chair
* Description: EPP run over QUIC
* Reference: This document

9. Implementation Status

Note to RFC Editor: Please remove this section and the reference to RFC 7942 [RFC7942] before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942 [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information

presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942 [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

9.1. Verisign EPP SDK

Organization: Verisign Inc.

Name: Verisign EPP SDK

Description: The Verisign EPP SDK includes both a full client implementation and a full server stub implementation of this specification.

Level of maturity: Development

Coverage: All aspects of the protocol are implemented with QUIC V1.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

URL: https://www.verisign.com/en_US/channel-resources/domain-registry-products/epp-sdks

10. Security Considerations

EPP over QUIC provides the similar security with EPP over TCP with TLS. Some related security issues have been discussed in [RFC5734] and [RFC9000].

EoQ servers run the risk of a resource exhaustion attack by allowing the creation of unlimited QUIC streams per QUIC connection. Servers SHOULD limit a client to a maximum number of QUIC streams per QUIC connection based on server capabilities and operational load.

11. Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions: Scott Hollenbeck Lucas Pardue, and Martin Thompson.

12. Normative References

[BCP14] Best Current Practice 14,
<https://www.rfc-editor.org/info/bcp14>.
At the time of writing, this BCP comprises the following:

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>.

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>.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[RFC5734] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Transport over TCP", STD 69, RFC 5734, DOI 10.17487/RFC5734, August 2009, <https://www.rfc-editor.org/info/rfc5734>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC7451] Hollenbeck, S., "Extension Registry for the Extensible Provisioning Protocol", RFC 7451, DOI 10.17487/RFC7451, February 2015, <https://www.rfc-editor.org/info/rfc7451>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[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/info/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/info/rfc9002>.

Appendix A. Change History

A.1. Change from 00 to 01

1. Added Dan Keathley and James Gould as co-authors and aligned the draft with EPP RFC 5734.

A.2. Change from 01 to 02

1. Make the clear distinction between an EPP connection and an EPP session for EoQ in the Session Management section.
2. Align the handling of the EPP <logout> command with RFC 5734, by including "A client MAY end an EoQ session by closing the QUIC stream" in the Session Management section.
3. Ensure that the relationship of the EoQ connection and the EoQ stream is maintained with the sentence "This means that a single QUIC connection may support multiple EoQ sessions" in the Session Management section.
4. Leverage the EoQ session in place of the more generic EPP session in the Message Exchange section.

A.3. Change from 02 to 03

1. Added the definition and use of the EoQ Connection Start Packet to explicitly trigger the creation of the QUIC stream and the EoQ connection to the server.
2. Added the Implementation Status section with the Verisign EPP SDK implementation.

A.4. draft-ietf-regext-epp-quic-00

1. updated to WG document

A.5. draft-ietf-regext-epp-quic-01

1. add section 8.1. EPP Extension Registry

A.6. draft-ietf-regext-epp-quic-02

Incorporated feedback from Lucas Pardue:

1. Added a list of terms in Section 2 "Conventions Used in This Document".
2. Changed ALPN "eoq" value to "eoq/0.1" to support versioning, which will be changed to "eoq/1.0" once passing WGLC.
3. Changed "A client MAY end an EoQ session by closing the QUIC stream" to "A client MAY end an EoQ session by closing the QUIC stream and the server MUST end the EoQ session by closing the QUIC stream".
4. Added language to Section 3 "Session Management" to make it clear that a bidirectional QUIC stream is client-initiated and inclusion of the "eoq/0.1" ALPN was added for the QUIC connection.
5. Added "QUIC supports four stream types, but EoQ only supports the client-initiated, bidirectional stream type." to Section 3 "Session Management" to be clear the stream types supported by EoQ.

A.7. draft-ietf-regext-epp-quic-03

Nit fixes, such as spelling fixes and small wording changes.

A.8. draft-ietf-regext-epp-quic-04

Incorporated feedback from Martin Thompson:

1. Changed ALPN "eoq/0.1" value to "EoQ" to match the value of "DoQ" for DNS over QUIC in RFC 9250, which doesn’t include versioning.
2. Added a reference to section 5 "Data Unit Format" in Section 7 for defining the packet framing of EoQ.
3. Address the additional pipelining considerations (independent command processing, batching, error in processing commands).

In the IANA Considerations section, removed the registration of the EoQ transport in the EPP Extension Registry.

Authors’ Addresses

Jiankang Yao
CNNIC
4 South 4th Street,Zhongguancun,Haidian District
Beijing
Beijing, 100190
China
Phone: +86 10 59116505
Email: yaojk@cnnic.cn

Hongtao Li
CNNIC
4 South 4th Street,Zhongguancun,Haidian District
Beijing
Beijing, 100190
China
Email: lihongtao@cnnic.cn

Man Zhang
CNNIC
4 South 4th Street,Zhongguancun,Haidian District
Beijing
Beijing, 100190
China
Email: zhangman@cnnic.cn

Daniel Keathley
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America
Email: dkeathley@verisign.com
URI: http://www.verisigninc.com

James Gould
VeriSign, Inc.
12061 Bluemont Way
Reston, VA 20190
United States of America
Email: jgould@verisign.com
URI: http://www.verisigninc.com