Stream: Internet Engineering Task Force (IETF)
RFC: 10087
Category: Standards Track
Published: November 2026
ISSN: 2070-1721
Authors: J. Yao, *CNNIC*; H. Li, *CNNIC*; M. Zhang, *CNNIC*; D. Keathley, *VeriSign, Inc.*; J. Gould, *VeriSign, Inc.*

RFC 10087
Extensible Provisioning Protocol (EPP) Transport over QUIC

Abstract

This document specifies how an Extensible Provisioning Protocol (EPP) session is mapped onto a QUIC connection. EPP over QUIC (EoQ) leverages features of the QUIC protocol.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc10087.

Copyright Notice

Copyright (c) 2026 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 3
2. Terminology 3
3. Session Management 4
4. Message Exchange 5
5. Data Unit Format 7
6. EoQ Connection Start Packet 7
7. Transport Considerations 8
8. Operational Considerations 9
8.1. Clients Fall Back with Management of Multiple Transport 9
8.2. Port Number Reuse 9
8.3. QUIC Support Announcement and Discovery 9
8.4. Configuration Parameters 9
8.5. Diagnostic and Troubleshooting 9
8.6. Address Validation 10
8.7. Authentication Considerations 10
8.8. 0-RTT and Session Resumption 10
8.9. MTU and Fragmentation 10
9. IANA Considerations 10
9.1. Registration of an EoQ Identification String 10
9.2. Registration of Port Number 11
10. Security Considerations 11
11. References 12
11.1. Normative References 12
11.2. Informative References 12
Acknowledgements 13
Authors' Addresses 13

1. Introduction

This document specifies how the Extensible Provisioning Protocol (EPP) [RFC5730] is mapped onto QUIC transport [RFC9000]. QUIC is a protocol that is based on UDP and incorporates 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.

EPP sessions use a single QUIC stream for all command and response exchanges throughout the session lifecycle. Unlike stateless transaction protocols that permit per-command independent streams, EPP is a stateful protocol with inherent sequential command dependencies defined in [RFC5730]. Maintaining a single stream preserves ordered transaction processing, consistent session state, and full compatibility with existing EPP operational and implementation models.

EPP [RFC5730] was designed for pluggability of transports that must follow Section 2.1 of [RFC5730]. EoQ defines a fully pluggable QUIC transport for EPP as an alternative to other EPP transports, such as EPP over TCP (EoT) [RFC5734]. The server can choose to offer EoQ in addition to EoT or as a replacement for EoT, based on the security and performance considerations of QUIC and the preferred deployment architecture of the server and clients.

EPP is primarily deployed for communication between domain registries and registrars, which commonly run many concurrent EPP sessions. Conventional EPP over TCP (EoT) requires a dedicated TCP connection for every single EPP session, resulting in connection overhead and resource consumption. EoQ addresses this issue by leveraging QUIC stream multiplexing: multiple independent EPP sessions can run over one single persistent QUIC connection.

2. Terminology

This document makes use of the following terms:

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" Application-Layer Protocol Negotiation (ALPN) [RFC7301] value. The EoQ connection maps to the client-server connection defined in Section 2.1 of [RFC5730], where the server returns an EPP <greeting>. A single 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 clients 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.

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.

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 [RFC7301] value and client-initiated, bidirectional QUIC streams. QUIC supports four stream types (Section 2.1 of [RFC9000]), but EoQ only supports the client-initiated, bidirectional stream type.

As shown in Figure 1, 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". By default, an EPP server **MUST** listen for QUIC connection requests on the well-known UDP port number updated by IANA (see Section 9.2), unless there is a mutual agreement to use another port number.

A successfully established QUIC connection is secured by the TLS support that QUIC provides using the "EoQ" 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 can be created by sending data". If 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 the same QUIC stream. After reading the EPP <greeting> message, and absent processing errors, 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 an EoQ connection. An authenticated QUIC stream, via a successful EPP <login>, corresponds to an EPP session, which is referred to as an 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 using the "EoQ" ALPN value in the cryptographic 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 the server closing the QUIC stream. A server **MAY** close EoQ sessions that have been open and active for longer than a server-defined limit. The max_idle_timeout transport parameter defined in Section 18.2 of [RFC9000] **MUST NOT** be set to a non-zero value by the client or server. Once the

last QUIC stream for a QUIC connection is closed, the server **MAY** either immediately end the QUIC connection or retain the open connection in accordance with local configuration and operational policy.

4. Message Exchange

EPP messages are passed on an EoQ connection, which is a client-initiated, bidirectional QUIC stream, defined in Section 3. 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. Absent such a limit, a server may be subject to overload and resource exhaustion. Section 4.6 of [RFC9000] discusses this situation.

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. For example, the time limit is 10 minutes for the server idle timeout and 8 minutes for client keep-alive. Absent local policy, a server **MUST** 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 an example of successful exchange over a single QUIC stream of a QUIC connection. Many QUIC streams may be opened and closed during the life of a QUIC connection.

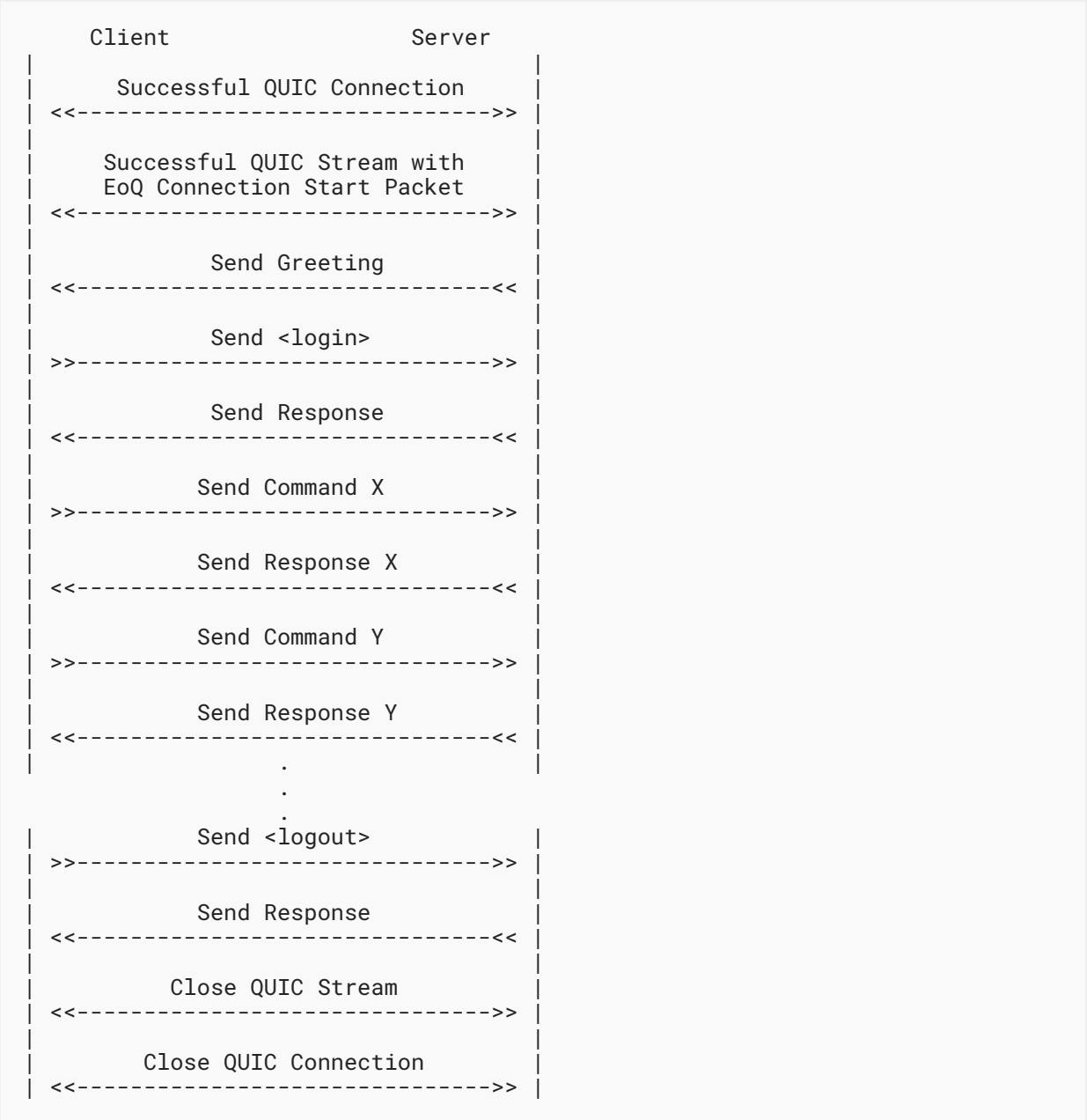


Figure 1: Example of Successful QUIC Client-Server Message Exchange

The EPP server **MUST** follow the "EPP Server State Machine" procedure (see Figure 1 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. The EPP data unit format is depicted in Figure 2 (one tick mark represents one bit position).

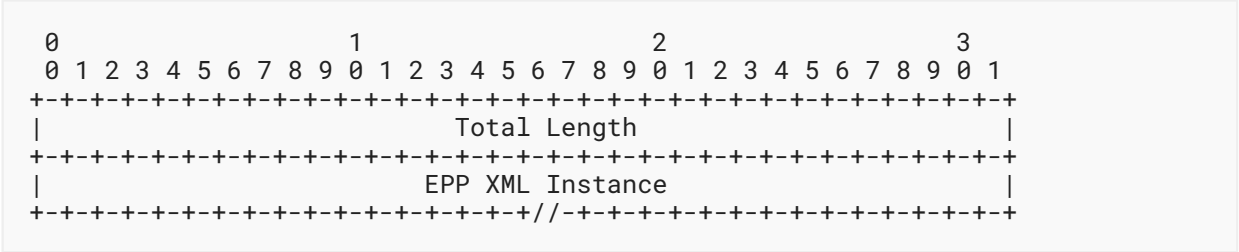


Figure 2: EPP Data Unit Format

The fields shown in Figure 2 are defined as follows:

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. The 32-bit total length matches the total length used in EoT, defined in [RFC5734], to support transport pluggability.

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 to the server to initialize the QUIC stream. Absent processing errors or local policy, 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 a valid data unit **MUST** be 24 octets, which includes 4 octets for the Total Length and 20 octets for the "EoQ Connection Start" constant value.

If the EoQ Connection Start Packet received by the server is not valid (e.g., when the total length is not exactly 24 octets), the server **MUST** close that QUIC stream. The server **MAY** return an error indication prior to closing the stream.

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 and features of this document as discussed below:

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. QUIC streams on top of QUIC connections, defined in Sections 2 and 5 of [RFC9000], are used to create stateful communications between the client and server.

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. Section 13 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.

  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. Operational Considerations

Some of the concerns examined within this section are also addressed in [RFC9308]. Authors of this specification encourage implementers to understand [RFC9308].

8.1. Clients Fall Back with Management of Multiple Transport

If the establishment of an EoQ connection fails, clients **MAY** attempt to fall back to EPP over TCP as specified in [RFC5734], depending on local deployment and security policy. It is up to clients to determine the mix of transports that best meets their business needs.

8.2. Port Number Reuse

Although [RFC5734] does only a request for TCP, the companion UDP port number was also allocated. That practice was prior to [RFC6335] when TCP and UDP port numbers were simultaneously assigned when either was requested. Section 9.2 updates EPP UDP/700 allocation to be used exclusively for EoQ. This update does not introduce any operational issues given that there are no known implementations of EPP over UDP.

8.3. QUIC Support Announcement and Discovery

There is no dedicated in-band mechanism defined in this specification for a server to explicitly announce EoQ support to clients. Operators **MAY** use out-of-band configuration or provisioning channels to advertise server EoQ support to clients in advance.

8.4. Configuration Parameters

Implementations **MAY** configure operational parameters to control EoQ session behavior. These parameters can include idle session timeout, maximum QUIC streams per connection, command processing timeout, and maximum session timeout. Servers and clients **SHOULD** configure limits to avoid session disruption and resource exhaustion.

> Implementation Note: There is no in-band protocol mechanism to align limits. Operators typically distribute standard settings via SDKs. Client timeout and stream limits shall be set lower than server values for graceful connection handling. Timeout rules are consistent across EoT and EoQ; per-connection stream limits are newly introduced in EoQ.

8.5. Diagnostic and Troubleshooting

Operators **SHOULD** log the EoQ connection establishment status, stream lifecycle events, and command transaction results for diagnostic purposes. QUIC transport errors and EPP protocol failures ought to be distinguishable to facilitate efficient troubleshooting. Implementations **MAY**

provide granular error reporting to help identify session termination, connection timeout, and stream closure root causes. Operators **MUST** redact sensitive data in the logs, such as user credentials and authorization information values.

8.6. Address Validation

EoQ implementations **MUST** follow the address validation requirements defined in Section 8 of [RFC9000] to mitigate potential amplification attacks and validate client address reachability.

8.7. Authentication Considerations

EoQ relies on the TLS authentication mechanisms defined in [RFC9001] for peer authentication and credential validation. Implementations **MUST** follow the same authentication practices specified for EPP over TLS in [RFC5734]. Server and client authentication procedures remain unchanged from existing EPP deployment models when running over QUIC transport.

8.8. 0-RTT and Session Resumption

Using 0-RTT for EoQ allows clients to establish connections and initiate EPP transactions without round-trip delay, enabling servers to use shorter idle timers and reduce connection overhead. Session resumption and 0-RTT introduce privacy and replay risks.

EPP commands fall into three categories: session management, queries, and transform commands. EPP query commands perform read-only object information retrieval operations. Only read-only EPP query commands **MAY** be carried in 0-RTT STREAM frames; all other EPP commands **MUST NOT** use 0-RTT. Clients **SHOULD** use resumption tickets only once specified in [RFC9846]. Servers **SHOULD** issue session-resumption tickets with a sufficiently long lifetime (e.g., 6 hours). Servers **SHOULD** implement the anti-replay mechanisms specified in Section 8 of [RFC9846].

EoQ implementations **SHOULD** follow the guidance in [RFC9846] and [RFC9001] to balance performance and risk mitigation. Failure to do so increases privacy exposure and replay attack risks.

8.9. MTU and Fragmentation

EoQ implementations **MUST** follow Section 14 of [RFC9000].

9. IANA Considerations

9.1. 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 under the "Transport Layer Security (TLS) Extensions" registry group available at <https://www.iana.org/assignments/tls-extensiontype-values/>.

Protocol: EoQ

Identification Sequence: 0x45 0x6F 0x51 ("EoQ")

Reference: RFC 10087

9.2. Registration of Port Number

The "Service Name and Transport Protocol Port Number Registry" (<https://www.iana.org/assignments/service-names-port-numbers/>) contains an entry for EPP UDP/700. However, there is no known usage of EPP over UDP. This document therefore specifies EPP over UDP as defined here. IANA has updated the description field of the EPP UDP/700 entry to be "EPP running over QUIC" and added this document as an additional reference alongside [RFC5734].

Service Name: epp

Port Number: 700

Transport Protocol(s): UDP

Assignee: IESG

Contact: IETF Chair

Description: EPP running over QUIC

Reference: [RFC5734] RFC 10087

10. Security Considerations

EPP over QUIC provides similar security foundations as EPP over TCP with TLS, though the security properties of QUIC specified in Section 21.1 of [RFC9000] differ from those of TLS/TCP in that QUIC is an encrypted transport protocol. Some related security issues are discussed in [RFC5734] and [RFC9000]. Sections 8.7 and 8.8 of this document also discuss related security issues.

EoQ servers run the risk of a resource exhaustion attack by allowing the creation of unlimited QUIC streams per QUIC connection. As indicated in Section 8.4, servers should limit a client to a maximum number of QUIC streams per QUIC connection based on server capabilities and operational load. Absent such a limit, the server may be subject to overload that would exhaust its resources.

The 32-bit total length field defined in Section 5 allows a declared payload size up to approximately 4 gigabytes. Implementations **MUST** validate the advertised total length against a locally configured maximum frame size limit before allocating memory or attempting to read the full frame. Without this check, an attacker could send a crafted frame with an excessively large length value to exhaust receiver memory and trigger a denial-of-service condition.

11. References

11.1. 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/info/rfc2119>.

[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>.

[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>.

[RFC9846] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 9846, DOI 10.17487/RFC9846, July 2026, <https://www.rfc-editor.org/info/rfc9846>.

[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>.

11.2. Informative References

[RFC9308] Kühlewind, M. and B. Trammell, "Applicability of the QUIC Transport Protocol", RFC 9308, DOI 10.17487/RFC9308, September 2022, <https://www.rfc-editor.org/info/rfc9308>.

Acknowledgements

The authors wish to thank the following persons for their feedback and suggestions: Scott Hollenbeck, Lucas Pardue, Martin Thomson, Mohamed Boucadair, Gorry Fairhurst, and Mahesh Jethanandani.

Authors' Addresses

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

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

Man Zhang
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
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