Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 7 May 2026

M. Loffredo
L. Luconi Trombacchi
M. Martinelli
IIT-CNR/Registro.it
D. Keathley
J. Gould
VeriSign, Inc.

3 November 2025

Extensible Provisioning Protocol (EPP) Transport over HTTPS
draft-ietf-regext-epp-https-02

Abstract

This document describes how an Extensible Provisioning Protocol (EPP) connection is mapped onto a Hypertext Transfer Protocol (HTTP) session. EPP over HTTP (EoH) requires the use of Transport Layer Security (TLS) to secure EPP information (i.e. HTTPS).

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 7 May 2026.

Table of Contents

1. Introduction

This document describes how EPP [RFC5730] is mapped onto HTTP [RFC9110]. Note that there are several versions of HTTP currently in use, including: HTTP/1.1 [RFC9112], HTTP/2 [RFC9113], and HTTP/3 [RFC9114]. As the differences among such versions do not affect the EPP mapping described in this document, hereinafter the version number is omitted except for presenting the special features in the underlying layers of the HTTP stack.

HTTP represents a higher-level abstraction of a network connection, removing the need to directly deal with all of the lower-level details of transport protocols. This makes HTTP much more compatible with cloud-native infrastructures, and facilitates faster development times and reduced maintenance costs in such environments.

Security services beyond those defined in EPP are provided by TLS via HTTPS Section 4.2.2 of [RFC9110].

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 HTTP is accomplished using existing HTTP methods, namely GET and POST. An EPP session exists on top of an EPP 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 implementing this specification MUST listen for HTTP session requests on a standard HTTP port assigned by IANA.

Even though HTTP itself is stateless, a stateful EPP session can be achieved using the mechanism described in [RFC6265]. This mechanism uses "Set-Cookie" and "Cookie" HTTP headers to facilitate a stateful HTTP session. Such a session is initiated by the client via sending a GET request to the sever. The GET request MUST include "application/epp+xml" (Appendix B of [RFC5730]) in the "Accept" HTTP header. The server MUST include the EPP Greeting in the response. The EPP Greeting must include "application/epp+xml" in the "Content-Type" header with the character encoding of the EPP XML (e.g., "application/epp+xml;charset=UTF-8") and MUST include ""no-cache" in the "Cache-Control" header and include "0" in the "Expires" header to disable caching. The server MUST use the "Set-Cookie" header to include a token that represents the identifier of the HTTP session. All subsequent HTTP requests that include the HTTP session identifier in the "Cookie" header will be treated as part of the session. The HTTP session represents an EPP connection, referred to as an EPP over HTTP (EoH) connection, which is initiated by the initial GET request.

The EPP session begins with a successful EPP <login> command on the EoH connection and can be referred to as an EPP over HTTP (EoH) 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. A server MAY also end an EPP session that has been either active or inactve for longer than a server-defined period. A server MAY end the HTTP session after ending the EPP session.

4. Message Exchange

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. With the exception of the EPP Greeting, EPP messages are initiated by the EPP client in the form of EPP commands. An EPP client MUST send all commands as HTTP POST requests (Section 6.4 of [RFC9110]). Each POST request MUST include the HTTP session identifier in the "Cookie" header and "application/epp+xml" in the "Accept" header. An EPP server MUST return an EPP response to an EPP command in the HTTP response to the respective HTTP request. The EPP command and the EPP response MUST include "application/epp+xml" in the "Content-Type" header with the character encoding of the EPP XML (e.g., "application/ epp+xml;charset=UTF-8"). The EPP response MUST include "no-cache" in the "Cache-Control" header and include "0" in the "Expires" header to disable caching.

HTTP does not guarantee that POST requests are idempotent. However, the semantics of EPP do require EPP commands to be idempotent, so processing a command more than once will produce the same net effect on the repository as successfully processing the command once.

The EPP command XML is framed by the content of the HTTP POST request, and the EPP response XML is framed by the content of the HTTP response. Each HTTP request MUST contain a single EPP message, and each HTTP response MUST contain a single EPP response. Commands MUST be processed independently and in the same order as received from the client.

Servers MUST NOT use HTTP return codes to signal clients about the failure of the EPP commands. The HTTP code 200 MUST be used for both successful and unsuccessful EPP requests. Servers MUST use HTTP codes to signal clients about the failure of the HTTP requests.

Servers MUST return an EPP 2002 response (i.e. Command use error) if the client issues an EPP command with either an empty or an invalid HTTP session identifier.

A server SHOULD impose a limit on the amount of time required for a client to issue a well-formed EPP command. A server SHOULD end an EPP session if a well-formed command is not received within the time limit.

HTTP/2 and HTTP/3 support a multiplexing feature that was introduced to address head-of-line blocking issues in previous HTTP versions. In the context of multiple requests being sent on a single HTTP connection, multiplexing allows the delivery of responses in a different order from how the requests were made. Due to this behavior, pipelining MUST NOT be used by EoH clients. EoH clients MUST wait for a server response to a command before sending a subsequent command.

A general state machine for an EPP server is described in Section 2 of [RFC5730]. A general client-server message exchange using HTTP is illustrated in Figure 1.

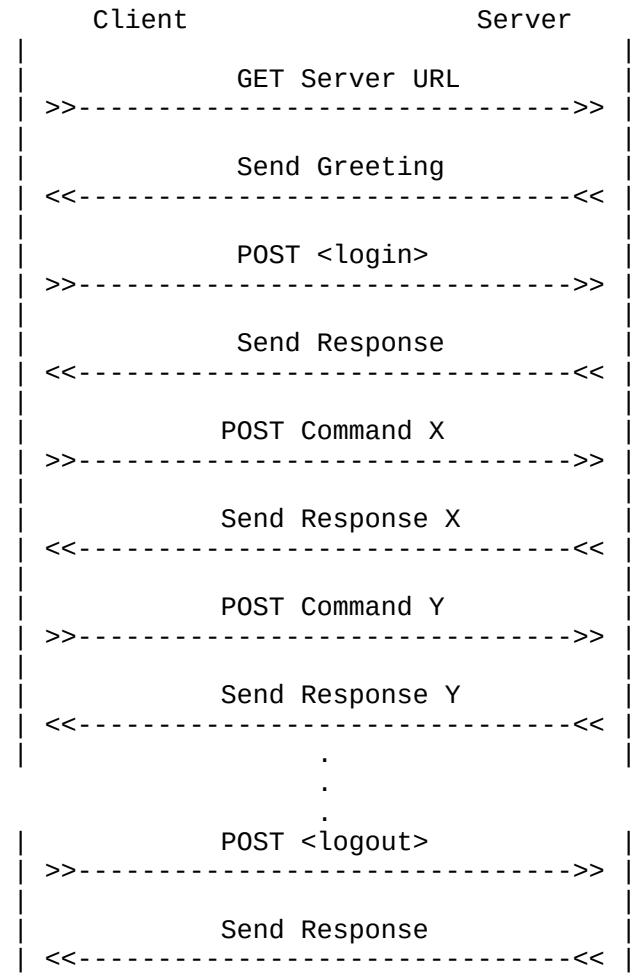

Figure 1: HTTP Client-Server Message Exchange

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

5. 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 HTTP protocol itself and features of this document.

* Command Order: Section 4 includes a requirement for ordered message delivery.

* Session Mapping: EPP session management is described in Section 3 of this document.

* Stateful Nature: Achieving the stateful nature of EPP is described in Section 3.

* Frame Data Units: Section 4 of this document describes how each EPP command is framed within the content of HTTP requests and responses.

* Congestion Avoidance: Section 3.9.3 of [RFC8095] confirms congestion avoidance as a feature of HTTP.

* Reliability: Section 3.9.3 of [RFC8095] confirms reliable message delivery as a feature of HTTP.

* Pipelining: Section 4 of this document stipulates that command pipelining must not be used in EoH.

6. IANA Considerations

6.1. EPP Extension Registry

The EPP transport described in this document should be registered by IANA in the "Extensions for the Extensible Provisioning Protocol (EPP)" registry described in RFC 7451 [RFC7451]. The details of the registration are as follows:

Name of Extension: "Extensible Provisioning Protocol (EPP) Transport over HTTPS"
Document status: Standards Track
Reference: (This specification)
Registrant Name and Email Address: IESG, <iesg@ietf.org>
Top-Level Domains(TLDs): Any
IPR Disclosure: None
Status: Active
Notes: None

7. Implementation Status

Note to RFC Editor: Please remove this section and the reference to [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 [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 [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".

7.1. 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. Both HTTP/1.1 and HTTP/2 were implemented, but HTTP/3 was not due to the lack of support of the underlying library.

Level of maturity: Development

Coverage: All aspects of the protocol are implemented with HTTP/1.1 and HTTP/2.

Licensing: GNU Lesser General Public License

Contact: jgould@verisign.com

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

7.2. IIT-CNR/Registro.it

Organization: Institute of Informatics and Telematics of National Research Council (IIT-CNR)/Registro.it

Name: .it EPP client and server

Description: This specification has been partially implemented on both the client and server sides. A slightly different implementation, which initiates the HTTP session upon completion of

an EPP Login request, has been running on the live platform since 2009. Registro .it is currently working to release a fully compliant implementation to the public test environment.

Level of Maturity: This is an implementation running in the live platform.

Coverage: This implementation includes all the functionality described in this specification, except that the HTTP session begins after an EPP Login request has been successfully processed.

Contact Information: Mario Loffredo, mario.loffredo@iit.cnr.it

8. Security Considerations

Since client credentials are included in the EPP <login> command, HTTPS (Section 4.2.2 of [RFC9110]) MUST be used to protect them from disclosure while in transit. HTTPS indicates that TLS is being used to secure the HTTP connection between the client and server. Transferring over TLS also prevents sniffing the HTTP session identifier and, consequently, impersonating a client to perform actions on registrars' objects. Servers are REQUIRED to support TLS 1.2 [RFC8446][RFC9155] or higher.

Servers are RECOMMENDED to implement additional measures to verify the client. These measures include IP allow-listing and locking the HTTP session identifier to the client's IP address.

HTTP session identifiers SHOULD be randomly generated to mitigate the risk of obtaining a valid one through a brute-force search. A HTTP session identifier SHOULD be at least 128 bits or 16 bytes long. An example of a reliable HTTP session identifier is the Universally Unique Identifier (UUID). Servers MAY limit the lifetime of active sessions to avoid them being exchanged for a long time.

The following measures MAY also be taken to control cookies usage:

* Restricting their scope through the "Domain" and "Path" attributes

* Limiting their lifetime through the "Max-Age" and "Expire" attributes

Other attributes that are normally used to secure the cookies and prevent them to be accessed by unintended parties or scripts, such as "HttpOnly" and "Secure", are meaningless in this context. Finally, servers are RECOMMENDED to perform additional checks to limit the rate of open EPP sessions and HTTP connections to mitigate the risk of congestion of requests. Here again, IP allow-listing could also be implemented to prevent DDoS attacks.

As a further measure to enforce the security, servers MUST require clients to present a digital certificate. Clients who possess and present a valid X.509 digital certificate, issued by a recognized Certification Authority (CA), could be identified and authenticated by a server who trusts the corresponding CA. This certificate-based mechanism is supported by HTTPS and can be used with EPP over HTTP.

If the EPP server is configured as a load balancer routing the requests to a pool of backend servers, some of the aforementioned checks SHOULD be implemented on the load balancer side.

9. Acknowledgements

The authors wish to acknowledge the input from the .IT technical team.

10. References

10.1. 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>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <https://www.rfc-editor.org/info/rfc6265>.

[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>.

[RFC8095] Fairhurst, G., Ed., Trammell, B., Ed., and M. Kuehlewind, Ed., "Services Provided by IETF Transport Protocols and Congestion Control Mechanisms", RFC 8095, DOI 10.17487/RFC8095, March 2017, <https://www.rfc-editor.org/info/rfc8095>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9112] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <https://www.rfc-editor.org/info/rfc9112>.

[RFC9113] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/info/rfc9113>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

[RFC9155] Velvindron, L., Moriarty, K., and A. Ghedini, "Deprecating MD5 and SHA-1 Signature Hashes in TLS 1.2 and DTLS 1.2", RFC 9155, DOI 10.17487/RFC9155, December 2021, <https://www.rfc-editor.org/info/rfc9155>.

10.2. Informative References

[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>.

Appendix A. Change History

A.1. Change from 02 to 03

1. Added Dan Keathley and James Gould as co-authors.

A.2. Change from 03 to 04

1. Clarified the difference between an EoH connection and an EPP session.
2. Added inclusion of the "Content-Type" header in every request and response except for the initial GET establishing the EoH connection.
3. Revised the Security Considerations section.

A.3. Change from 04 to 05

1. Added the Implementation Status section with the Verisign EPP SDK implementation.
2. Removed the "Internationalization Considerations" section and added the EoH character encoding matching the EPP XML character encoding for the EPP Greeting, the EPP commands, and the EPP responses.
3. Added inclusion of the "Cache-Control" and "Expires" headers for the the EPP Greeting, the EPP commands, and the EPP responses.

A.4. Change from regext 00 to regext 01

1. Added EPP Extension Registry registration for the EPP transport over HTTPS.

A.5. Change from regext 01 to regext 02

1. Added IIT-CNR/Registro.it implementation.

Authors' Addresses

Mario Loffredo
IIT-CNR/Registro.it
Via Moruzzi,1
56124 Pisa
Italy
Email: mario.loffredo@iit.cnr.it
URI: https://www.iit.cnr.it

Lorenzo Luconi Trombacchi
IIT-CNR/Registro.it
Via Moruzzi,1
56124 Pisa
Italy
Email: lorenzo.luconi@iit.cnr.it
URI: https://www.iit.cnr.it

Maurizio Martinelli
IIT-CNR/Registro.it
Via Moruzzi,1
56124 Pisa
Italy
Email: maurizio.martinelli@iit.cnr.it
URI: https://www.iit.cnr.it

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