XEP-0246: End-to-End XML Streams

Peter Saint-Andre
mailto:peter@andyet.net
xmpp:stpeter@stpeter.im
https://stpeter.im/

2016-01-20
Version 0.1.1

Status	Type	Short Name
Deferred	Standards Track	NOT_YET_ASSIGNED

This specification defines methods for communicating via end-to-end XML streams over a logical or physical connection that provides a reliable transport between two endpoints.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction 1

2 Initiating an e2e Stream 1

3 Stream Encryption 2

4 Exchanging Stanzas 3

5 Ending an e2e Stream 3

6 Security Considerations 4

7 IANA Considerations 4

8 XMPP Registrar Considerations 4

1 Introduction

XMPP as defined in XMPP Core [1] does not support direct interaction between endpoints, since it requires a client to authenticate an XML stream with a ”home” server and send of all of its outbound XML stanzas through that server (which potentially can route those stanzas through a peer server for delivery to the intended recipient). However, in some scenarios it is desirable to establish end-to-end XML streams between two endpoints instead of relying on the standard client-server architecture. These scenarios include:

- Two endpoints cannot access an XMPP server
- Two endpoints want to enforce end-to-end encryption
- Two endpoints want to send a high volume of XMPP traffic but the intermediate servers enforce rate limits

The first scenario is addressed by Link-Local Messaging (XEP-0174) [2]. The second and third scenarios are addressed by Jingle XML Streams (XEP-0247) [3]. Both of those technologies result in the establishment of a direct or mediated connection between two endpoints, such as a direct TCP connection, a bytestream through SOCKS5 (SOCKS5 Bytestreams (XEP-0065) [4]) or XMPP itself (In-Band Bytestreams (XEP-0047) [5]), or other future transport methods such as RFC 6544 [6].
Once two endpoints have opened a direct or mediated connection, they can establish an XML stream over that connection for end-to-end ”(”e2e”) communication. We call this an ”e2e stream”.

2 Initiating an e2e Stream

The initiator and recipient essentially follow the process defined in RFC 6120 to establish XML streams between themselves.
First, the initiator opens an XML stream to the recipient over the negotiated transport.

Listing 1: Opening a Stream

```
<stream:stream
        xmlns='jabber:client'
        xmlns:stream='http://etherx.jabber.org/streams'
```

[1] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[2] XEP-0174: Link-Local Messaging <https://xmpp.org/extensions/xep-0174.html>.
[3] XEP-0247: Jingle XML Streams <https://xmpp.org/extensions/xep-0247.html>.
[4] XEP-0065: SOCKS5 Bytestreams <https://xmpp.org/extensions/xep-0065.html>.
[5] XEP-0047: In-Band Bytestreams <https://xmpp.org/extensions/xep-0047.html>.
[6] RFC 6544: TCP Candidates with Interactive Connectivity Establishment (ICE) <http://tools.ietf.org/html/rfc6544>.

```
        from='romeo@forza'
        to='juliet@pronto'
        version='1.0'>
```

In accordance with RFC 6120, the initial stream header SHOULD include the 'to' and 'from' attributes. In the case of XEP-0174, these SHOULD be the username@machine-name advertised in the PTR record. In the case of Jingle XML Streams, these SHOULD be the bare JIDs (<localpart@domain.tld> or <domain.tld>) of the entities as communicated via XMPP.
If the initiator supports stream features and the other stream-related aspects of XMPP 1.0 as specified in RFC 6120, then it SHOULD include the version='1.0' flag as shown in the previous example.
The recipient then responds with a stream header as well:

Listing 2: Stream Header Response

```
<stream:stream
        xmlns='jabber:client'
        xmlns:stream='http://etherx.jabber.org/streams'
        from='juliet@pronto'
        to='romeo@forza'
        version='1.0'>
```

If both the initiator and recipient included the version='1.0' flag, the recipient SHOULD also send stream features as specified in RFC 6120:

Listing 3: Recipient Sends Stream Features

```
<stream:features>
  <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
</stream:features>
```

3 Stream Encryption

The mere exchange of stream headers results in an unencrypted and unauthenticated channel between the two entities. The entities SHOULD upgrade the channel to an encrypted stream using the XMPP STARTTLS command defined in XMPP Core [7] using RFC 5246 [8], optionally followed by SASL negotiation for mutual authentication (see RFC 4422 [9]).
End-to-end XML streams can be negotiated between two XMPP clients, between an XMPP client and a remote XMPP service (i.e., a service with which a client does not have a direct XML stream, such as a remote Multi-User Chat (XEP-0045) [10] room), or between two remote XMPP services. Therefore, if standard X.509 certificates are used then a party to an e2e XML

[7] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[8] RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 <http://tools.ietf.org/html/rfc5246>.
[9] RFC 4422: Simple Authentication and Security Layer (SASL) <http://tools.ietf.org/html/rfc4422>.
[10] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.

stream will present either a client certificate or a server certificate as appropriate. If X.509 certificates are used, they MUST at a minimum be generated and validated in accordance with the certificate guidelines guidelines provided in RFC 6120 [11]; however, applications of end-to-end XML streams MAY define supplemental guidelines for certificate validation in the context of particular architectures, such as XEP-0174 for link-local streams and XEP-0247 for direct or mediated streams negotiated through XMPP servers.
To ease the transition from the PGP-based object encryption method specified in Current Jabber OpenPGP Usage (XEP-0027) [12], clients using TLS for e2e streams MAY use the OpenPGP TLS extension defined in RFC 5081 [13] (if available).
Use of other TLS extensions MAY be appropriate as well, including those defined in RFC 5246 [14] and RFC 5054 [15].

4 Exchanging Stanzas

Once the streams are established, either entity then can send XMPP message, presence, and IQ stanzas, with or without 'to' and 'from' addresses.

Listing 4: Sending a Message

```
<message from='romeo@forza' to='juliet@pronto'>
  <body>M'lady, I would be pleased to make your acquaintance.</body>
</message>
```

Listing 5: A Reply

```
<message from='juliet@pronto' to='romeo@forza'>
  <body>Art thou not Romeo, and a Montague?</body>
</message>
```

5 Ending an e2e Stream

To end the stream, either party closes the XML stream:

Listing 6: Closing the Stream

```
</stream:stream>
```

[11] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[12] XEP-0027: Current Jabber OpenPGP Usage <https://xmpp.org/extensions/xep-0027.html>.
[13] RFC 5081: Using OpenPGP Keys for Transport Layer Security (TLS) Authentication <http://tools.ietf.org/html/rfc5081>.
[14] RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 <http://tools.ietf.org/html/rfc5246>.
[15] RFC 5054: Using the Secure Remote Password (SRP) Protocol for TLS Authentication <http://tools.ietf.org/html/rfc5054>.

The other party then closes the stream in the other direction as well:

Listing 7: Closing the Stream

```
</stream:stream>
```

Both parties then SHOULD close the logical or physical connection between them.

6 Security Considerations

End-to-end streams SHOULD be encrypted; see the Stream Encryption section of this document.

7 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [16].

8 XMPP Registrar Considerations

This document requires no interaction with the XMPP Registrar [17].

[16] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

[17] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.