



XMPP

XEP-0247: Jingle XML Streams

Peter Saint-Andre
<mailto:stpeter@stpeter.im>
<xmpp:stpeter@jabber.org>
<https://stpeter.im/>

Justin Karneges
<mailto:justin@karneges.com>
<xmpp:justin@andbit.net>

Dirk Meyer
<mailto:dmeyer@tzi.de>
<xmpp:dmeyer@jabber.org>

2009-02-20
Version 0.2

Status	Type	Short Name
Deferred	Standards Track	NOT_YET_ASSIGNED

This specification defines a Jingle application type for establishing direct or mediated XML streams between two entities over any streaming transport. This technology thus enables two entities to establish a trusted connection for end-to-end encryption or for bypassing server limits on large volumes of XMPP traffic.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2024 by the [XMPP Standards Foundation](#) (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the "Specification"), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF's Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy>) or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1	Introduction	1
2	Protocol	1
3	Implementation Notes	7
3.1	Mandatory to Implement Technologies	7
3.2	Preference Order of Transport Methods	7
4	IANA Considerations	8
5	XMPP Registrar Considerations	8
5.1	Protocol Namespaces	8
5.2	Protocol Versioning	8
5.3	Jingle Application Formats	8
6	XML Schema	9

1 Introduction

The standard client-server architecture for XMPP communication provides a stable infrastructure for real-time communication. However, there are certain situations in which it is desirable to bypass the standard client-server architecture, including:

- 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 situation is addressed by [Link-Local Messaging \(XEP-0174\)](#)¹. However, if the endpoints already have client-to-server connections but wish to bypass those connections or leverage those streams for a higher-level application such as end-to-end encryption, it is desirable for the two endpoints to negotiate an end-to-end XML stream. This specification defines methods for doing so, where the application format is an XML stream and the transport method is any direct or mediated streaming transport, such as [Jingle In-Band Bytestreams Transport Method \(XEP-0261\)](#)² (mediated), [Jingle SOCKS5 Bytestreams Transport Method \(XEP-0260\)](#)³ (direct or mediated), or a future ice-tcp Jingle transport (direct or mediated) based on [RFC 6544](#)⁴.

2 Protocol

This section provides a friendly introduction to Jingle XML streams.

First, the party that wishes to initiate the stream determines the responder's capabilities (via [Service Discovery \(XEP-0030\)](#)⁵ or [Entity Capabilities \(XEP-0115\)](#)⁶). Here we assume that the responder supports a service discovery feature of 'urn:xmpp:jingle:apps:xmlstream:0' (see [Namespace Versioning](#) regarding the possibility of incrementing the version number) corresponding to the Jingle XML stream functionality defined herein, as well as the 'urn:xmpp:jingle:transports:ibb:0' feature.

The initiator then sends a Jingle session-initiation request to the responder. The content-type of the request specifies three things:

1. An application type of "urn:xmpp:jingle:apps:xmlstream:0".

¹XEP-0174: Link-Local Messaging <<https://xmpp.org/extensions/xep-0174.html>>.

²XEP-0261: Jingle In-Band Bytestreams Transport Method <<https://xmpp.org/extensions/xep-0261.html>>.

³XEP-0260: Jingle SOCKS5 Bytestreams Transport Method <<https://xmpp.org/extensions/xep-0260.html>>.

⁴RFC 6544: TCP Candidates with Interactive Connectivity Establishment (ICE) <<http://tools.ietf.org/html/rfc6544>>.

⁵XEP-0030: Service Discovery <<https://xmpp.org/extensions/xep-0030.html>>.

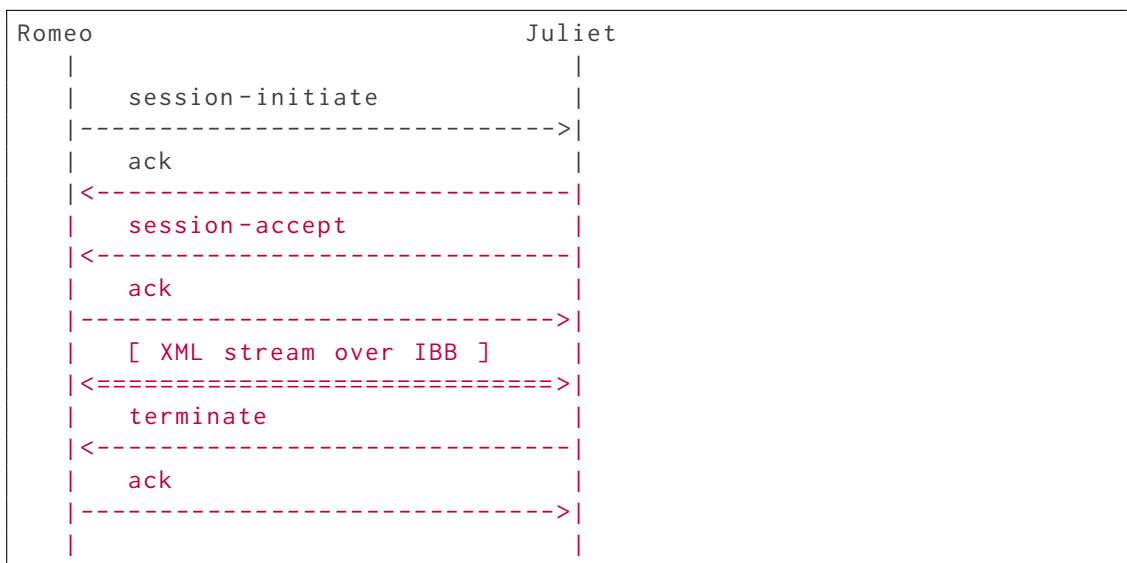
⁶XEP-0115: Entity Capabilities <<https://xmpp.org/extensions/xep-0115.html>>.

- Options for the streaming transport method, such as In-Band Bytestreams ("IBB") as defined in [Jingle In-Band Bytestreams Transport Method \(XEP-0261\)](#)⁷ or SOCKS5 Bytestreams ("S5B") as defined in XEP-0260.

Note: It is STRONGLY RECOMMENDED to encrypt all end-to-end XML streams as described in Jingle-XTLS (currently located at <http://xmpp.org/extensions/inbox/jingle-xtls.html>). Those security flows are NOT described here.

In this example, the initiator is <romeo@montague.lit>, the responder is <juliet@capulet.lit>, and the initiation request specifies a transport method of "jingle-ibb" (i.e., [Jingle In-Band Bytestreams Transport Method \(XEP-0261\)](#)⁸).

The flow is as follows.



First the initiator sends a Jingle session-initiate.

Listing 1: Initiator sends session-initiate

```

<iq from='romeo@montague.lit/orchard'
  id='ty1bf726'
  to='juliet@capulet.lit/balcony'
  type='set'>
  <jingle xmlns='urn:xmpp:jingle:0'
    action='session-initiate'
    initiator='romeo@montague.lit/orchard'
    sid='851ba2'>
    <content creator='initiator' name='xmlstream'>
      <description xmlns='urn:xmpp:jingle:apps:xmlstream:0' />
  
```

⁷XEP-0261: Jingle In-Band Bytestreams Transport Method <https://xmpp.org/extensions/xep-0261.html>.

⁸XEP-0261: Jingle In-Band Bytestreams Transport Method <https://xmpp.org/extensions/xep-0261.html>.

```

    <transport xmlns='urn:xmpp:jingle:transports:ibb:0'
              block-size='4096'
              sid='vj3hs98y' />
  </content>
</jingle>
</iq>

```

The responder immediately acknowledges receipt of the Jingle session-initiate.

Listing 2: Responder acknowledges session-initiate

```

<iq from='juliet@capulet.lit/balcony'
    id='ty1bf726'
    to='romeo@montague.lit/orchard'
    type='result' />

```

If the responding user accepts the session then her client sends a session-accept.

Listing 3: Responder sends session-accept

```

<iq from='juliet@capulet.lit/balcony'
    id='hwd987h'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:0'
          action='session-accept'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkl37jfea'>
    <content creator='initiator' name='stub'>
      <description xmlns='urn:xmpp:jingle:apps:stub:0' />
      <transport xmlns='urn:xmpp:jingle:transports:ibb:0'
                  block-size='4096'
                  sid='vj3hs98y' />
    </content>
  </jingle>
</iq>

```

The initiator acknowledges receipt.

Listing 4: Initiator acknowledges session-accept

```

<iq from='romeo@montague.lit/orchard'
    id='hwd987h'
    to='juliet@capulet.lit/balcony'
    type='result' />

```

The clients can then begin to exchange XMPP data over the in-band bytestream. Because the transport is an in-band bytestream, the XMPP data is prepared as described in [In-Band](#)

Bytestreams (XEP-0047)⁹ (i.e., Base64-encoded).

First the initiator sends an initial stream header to the responder.

Listing 5: Initial stream header (unencoded)

```
<stream:stream
  xmlns='jabber:client'
  xmlns:stream='http://etherx.jabber.org/streams'
  from='romeo@montague.lit/orchard'
  to='juliet@capulet.lit/balcony'
  version='1.0'>
```

Note: In accordance with XMPP IM¹⁰, the initial stream header SHOULD include the 'to' and 'from' attributes, which SHOULD specify the full JIDs of the clients. If the initiator supports stream features and the other stream-related aspects of XMPP 1.0 as specified in RFC 3920¹¹, then it SHOULD include the version='1.0' flag as shown in the previous example.

Listing 6: Initial stream header (encoded in IBB) and IQ-result

```
<iq from='romeo@montague.net/orchard'
  id='ur73n153'
  to='juliet@capulet.com/balcony'
  type='set'>
  <data xmlns='http://jabber.org/protocol/ibb' seq='0' sid='vj3hs98y'>
    PHN0cmVhbTpzdHJlYW0geG1sbnM9J2phYmJlcjpbG11bnQnIHhtbG5zOnN0cmVh
    bT0naHR0cDovL2V0aGVyeC5qYWJiZXIub3JnL3N0cmVhbXMnIGZyb209J3JvbWVv
    QG1vbnRhZ3V1LmxpdC9vcmlNoYXJkYyB0bz0nanVsaWV0QGNhcHVzZXQubGl0L2Jh
    bGNvbknIHZ1cnNpb249JzEuMCC+
  </data>
</iq>

<iq from='juliet@capulet.com/balcony'
  id='ur73n153'
  to='romeo@montague.net/orchard'
  type='result' />
```

The responder then sends a response stream header back to the initiator (because this stream header is sent in the other direction, the IBB 'seq' attribute has a value of zero, not 1).

Listing 7: Response stream header

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

⁹XEP-0047: In-Band Bytestreams <<https://xmpp.org/extensions/xep-0047.html>>.

¹⁰RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <<http://tools.ietf.org/html/rfc6121>>.

¹¹RFC 3920: Extensible Messaging and Presence Protocol (XMPP): Core <<http://tools.ietf.org/html/rfc3920>>.

```

from='juliet@capulet.lit/balcony'
id='hs91gh1836d8s717'
to='romeo@montague.lit/orchard'
version='1.0'>

```

Listing 8: Response stream header (encoded in IBB) and IQ-result

```

<iq from='juliet@capulet.com/balcony'
  id='pd61g397'
  to='romeo@montague.net/orchard'
  type='set'>
  <data xmlns='http://jabber.org/protocol/ibb' seq='1' sid='vj3hs98y'>
    PHN0cmVhbTpdHJlYW0geG1sbnM9J2phYmJlcjpbG11bnQnIHhtbG5zOnN0cmVh
    bT0naHR0cDovL2V0aGVyeC5qYWJiZXIub3JnL3N0cmVhbXMnIGZyb209J2p1bG1l
    dEBjYXB1bGV0LmxdC9iYWxjb255JyBpZD0naHM5MmWdoMTgzNmQ4czcxNycgdG89
    J3JvbWVvQG1vbRnRhZ3VlLmxdC9vcNoYXJkYyB2ZXJzaW9uPScxLjAnPg==
  </data>
</iq>

<iq from='romeo@montague.net/orchard'
  id='pd61g397'
  to='juliet@capulet.com/balcony'
  type='result' />

```

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

Listing 9: A message (unencoded)

```

<message from='romeo@montague.lit/orchard' to='juliet@capulet.lit/
  balcony'>
  <body>M&apos;lady, I would be pleased to make your acquaintance.</
  body>
</message>

```

Listing 10: A message (encoded in IBB) and IQ-result

```

<iq from='romeo@montague.net/orchard'
  id='iq7dh294'
  to='juliet@capulet.com/balcony'
  type='set'>
  <data xmlns='http://jabber.org/protocol/ibb' seq='1' sid='vj3hs98y'>
    PG11c3NhZ2UgZnJvbT0ncm9tZW9AbW9udGFndWUubG10L29yY2hhcmQnIHRvPSdq
    dWxpZXRAY2FwdWxldC5saXQvYmFsY29ueSc+PGJvZHK+TSZhcG9z02xhZHksIEkg
    d291bGQgYmUgcGx1YXNlZCB0byBtYWtlIH1vdXIgYWNxdWFpbnRhbml1LjwvYm9k
    eT48L211c3NhZ2U+
  </data>
</iq>

```



```
<iq from='juliet@capulet.com/balcony'
  id='iq7dh294'
  to='romeo@montague.net/orchard'
  type='result'/>
```

The responder could then send a reply.

Listing 11: A reply (unencoded)

```
<message from='juliet@capulet.lit/balcony' to='romeo@montague.lit/
  orchard'>
  <body>Art thou not Romeo, and a Montague?</body>
</message>
```

Listing 12: A reply (encoded in IBB) and IQ-result

```
<iq from='juliet@capulet.com/balcony'
  id='hr91hd63'
  to='romeo@montague.net/orchard'
  type='set'>
  <data xmlns='http://jabber.org/protocol/ibb' seq='1' sid='vj3hs98y'>
    PG11c3NhZ2UgZnJvbT0nanVsaWV0QGNhcHVzZXQubG10L2JhbGNvbknIHRvPSdy
    b211b0Btb250YWd1ZS5saXQvb3JjaGFyZCtPGJvZk+QXJ0IHRob3UgYm90IFJv
    bWVvLCBhbmQgYSBNb250YWd1ZT88L2JvZk+PC9tZXNzYWdlPg==
  </data>
</iq>

<iq from='romeo@montague.net/orchard'
  id='kr91n475'
  to='juliet@capulet.com/balcony'
  type='result'/>
```

To end the XML stream, either party sends a closing `</stream:stream>` element.

Listing 13: Stream close (unencoded)

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

Listing 14: Stream close (encoded in IBB) and IQ-result

```
<iq from='juliet@capulet.com/balcony'
  id='kr91n475'
  to='romeo@montague.net/orchard'
  type='set'>
  <data xmlns='http://jabber.org/protocol/ibb' seq='2' sid='vj3hs98y'>
    PC9zdHJlYW06c3RyZWFTPg==
  </data>
</iq>
```

```
<iq from='romeo@montague.net/orchard'  
  id='kr91n475'  
  to='juliet@capulet.com/balcony'  
  type='result'/>
```

However, even after the application-level XML stream is terminated, the negotiated Jingle transport (here in-band bytestream) continues and could be re-used. To completely terminate the Jingle session, the terminating party would then also send a Jingle session-terminate message.

Listing 15: Responder terminates the stream and session

```
<iq from='juliet@capulet.lit/balcony'  
  id='psy617r4'  
  to='romeo@montague.lit/orchard'  
  type='set'>  
  <jingle xmlns='urn:xmpp:jingle:0'  
    action='session-terminate'  
    initiator='romeo@montague.lit/orchard'  
    sid='851ba2' />  
</iq>
```

The other party then acknowledges the Jingle session-terminate.

Listing 16: Initiator acks session-terminate

```
<iq from='romeo@montague.lit/orchard'  
  id='psy617r4'  
  to='juliet@capulet.lit/balcony'  
  type='result'/>
```

3 Implementation Notes

3.1 Mandatory to Implement Technologies

An implementations MUST support the Jingle IBB Transport Method ([Jingle In-Band Bytestreams Transport Method \(XEP-0261\)](#)¹²) as a dependable method of last resort. An implementation SHOULD support other streaming transport methods as well, such as the Jingle S5B Transport Method (XEP-0260).

3.2 Preference Order of Transport Methods

An application MAY present transport methods in any order, except that the Jingle IBB Transport Method MUST be the lowest preference.

¹²XEP-0261: Jingle In-Band Bytestreams Transport Method <<https://xmpp.org/extensions/xep-0261.html>>.

4 IANA Considerations

No interaction with the [Internet Assigned Numbers Authority \(IANA\)](#) ¹³ is required as a result of this document.

5 XMPP Registrar Considerations

5.1 Protocol Namespaces

This specification defines the following XML namespace:

- `urn:xmpp:jingle:apps:xmlstream:0`

Upon advancement of this specification from a status of Experimental to a status of Draft, the [XMPP Registrar](#) ¹⁴ shall add the foregoing namespaces to the registry located at <https://xmpp.org/registrar/namespaces.html>, as described in Section 4 of [XMPP Registrar Function \(XEP-0053\)](#) ¹⁵.

5.2 Protocol Versioning

If the protocol defined in this specification undergoes a revision that is not fully backwards-compatible with an older version, the XMPP Registrar shall increment the protocol version number found at the end of the XML namespaces defined herein, as described in Section 4 of XEP-0053.

5.3 Jingle Application Formats

The XMPP Registrar shall include "xmlstream" in its registry of Jingle application formats. The registry submission is as follows:

¹³The 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/>.

¹⁴The 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/>.

¹⁵XEP-0053: XMPP Registrar Function <https://xmpp.org/extensions/xep-0053.html>.

```
<application>
  <name>xmlstream</name>
  <desc>Jingle sessions for an end-to-end XML stream</desc>
  <transport>streaming</transport>
  <doc>XEP-0247</doc>
</application>
```

6 XML Schema

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='urn:xmpp:jingle:apps:xmlstream:0'
  xmlns='urn:xmpp:jingle:apps:xmlstream:0'
  elementFormDefault='qualified'>

  <xs:element name='description' type='empty' />

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value='' />
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```