XEP-0177: Jingle Raw UDP Transport Method

Joe Beda
mailto:jbeda@google.com
xmpp:jbeda@google.com

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

Scott Ludwig
mailto:scottlu@google.com
xmpp:scottlu@google.com

Joe Hildebrand
mailto:jhildebr@cisco.com
xmpp:hildjj@jabber.org

Sean Egan
mailto:seanegan@google.com
xmpp:seanegan@google.com

2009-12-23
Version 1.1

Status	Type	Short Name
Draft	Standards Track	jingle-raw-udp

This specification defines a Jingle transport method that results in sending media data using raw datagram associations via the User Datagram Protocol (UDP). This simple transport method does not provide NAT traversal, and the ICE-UDP transport method should be used if NAT traversal is required.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Jingle (XEP-0166) [1] defines a framework for negotiating and managing out-of-band data sessions over XMPP. In order to provide a flexible framework, the base Jingle specification defines neither data transport methods nor application formats, leaving that up to separate specifications. The current document defines a transport method for establishing and managing data between XMPP entities using a raw User Datagram Protocol (UDP) association (see RFC 768 [2]). This "raw-udp" method results in a datagram transport method suitable for use in media applications where some packet loss is tolerable (e.g., audio and video).
The Raw UDP transport does not provide end-to-end traversal of Network Address Translators (NATs), or even basic connectivity checks; if NAT traversal is needed, Jingle clients SHOULD use RFC 5245 [3] as described in Jingle ICE-UDP Transport Method (XEP-0176) [4]. The Raw UDP transport method is defined only for the purpose of specifying the IP address and port that an entity considers "most likely to succeed" and is a "hit-or-miss" method that might work end-to-end in some circumstances (especially when the sending entity is a gateway or relay, for example when a back-to-back user agent or call manager sends an early media offer to the initiator on behalf of the responder, as described in Jingle RTP Sessions (XEP-0167) [5]).

2 Requirements

The Jingle transport method defined herein is designed to meet the following requirements:

1. Make it possible to establish and manage out-of-band connections between two XMPP entities over the IP address and port that the parties consider most likely to succeed.
2. Make it relatively easy to implement support in standard Jabber/XMPP clients.
3. Where communication with non-XMPP entities is needed, push as much complexity as possible onto server-side gateways between the XMPP network and the non-XMPP network.

3 Jingle Conformance

In accordance with Section 10 of XEP-0166, this document specifies the following information related to the Jingle Raw UDP transport type:

[1] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.
[2] RFC 768: User Datagram Protocol <http://tools.ietf.org/html/rfc0768>.
[3] RFC 5245: Interactive Connectivity Establishment (ICE) <http://tools.ietf.org/html/rfc5245>.
[4] XEP-0176: Jingle ICE-UDP Transport Method <https://xmpp.org/extensions/xep-0176.html>.
[5] XEP-0167: Jingle RTP Sessions <https://xmpp.org/extensions/xep-0167.html>.

1. The transport negotiation process is defined in the Protocol Description section of this document.

2. The semantics of the <transport/> element are defined in the Transport Initiation section of this document.

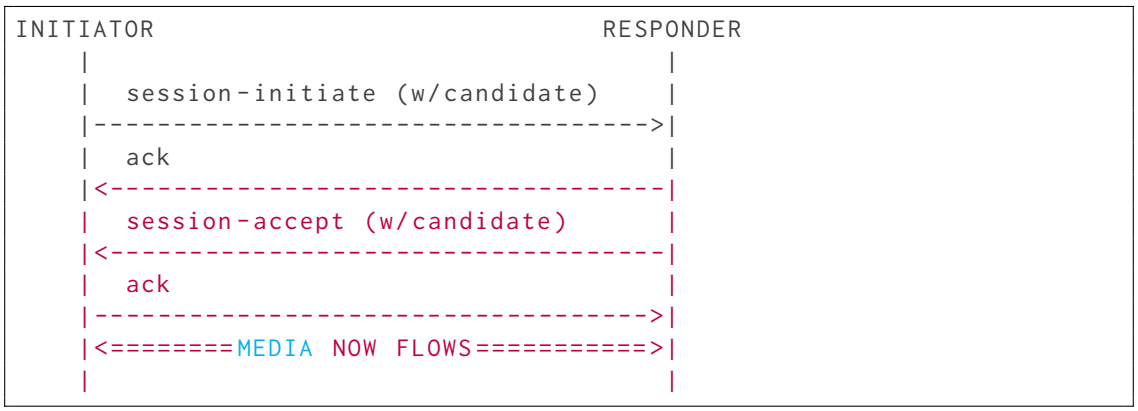
3. Successful negotiation of the Raw UDP method results in use of a datagram transport that is suitable for applications where some packet loss is tolerable, such as audio and video.

4. If multiple components are to be communicated by the application type that uses the transport, the transport shall support those components and assign identifiers for them as described in the specification that defines the application type.

4 Protocol Description

4.1 Flow

The overall protocol flow for negotiation of the Jingle Raw UDP Transport Method is as follows.



The following sections describe the protocol flow in detail.

4.2 Transport Initiation

In order for the initiator in a Jingle exchange to start the negotiation, it sends a Jingle ”session-initiate” stanza that includes at least one content type, as described in XEP-0166. If the initiator wishes to negotiate the Raw UDP transport for a given content type, it MUST include a <transport/> child element qualified by the ’urn:xmpp:jingle:transports:raw-udp:1’

namespace (see Namespace Versioning regarding the possibility of incrementing the version number), which MUST [6] include the initiator's Raw UDP candidate via the 'ip', 'port', 'generation', and 'id' attributes of the <candidate/> element. The <transport/> element MAY include more than one <candidate/> element (typically one for RTP and another for RTCP).

Listing 1: Initiation

```
<iq from='romeo@montague.lit/orchard'
    id='tp2hd816'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='voice'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
        <payload-type id='18' name='G729'/>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:raw-udp:1'>
        <candidate component='1'
                   generation='0'
                   id='a9j3mnbtu1'
                   ip='10.1.1.104'
                   port='13540'/>
      </transport>
    </content>
  </jingle>
</iq>
```

All attributes are REQUIRED. The 'ip' and 'port' attributes are self-explanatory. The 'component' attribute enables the parties to distinguish between different aspects of the media stream that each need to use a separate transport address (e.g., RTP and RTCP). The 'generation' attribute defines which version of this candidate is in force (this is useful if the candidate is redefined mid-stream, for example if the port is changed). The 'id' attribute uniquely identifies this candidate for tracking purposes.
Note: The "Raw UDP candidate" is the candidate that the entity has reason to believe will be most likely to succeed for that content type, and thus is equivalent to the "default" candidate as described in the ICE specification. This is not necessarily the entity's preferred address for communication, but instead is the "address most likely to succeed", i.e., the address that is assumed to be reachable by the vast majority of target entities. To determine reachability, the sender needs to classify ahead of time the permissiveness of the NAT or firewall it is behind, if any. It then SHOULD assign the Raw UDP candidate as follows, where the candidate types are as described in ICE:

[6] This is required to avoid a round trip and help expedite the negotiation.

NAT Type	Recommended Raw UDP Candidate Type
None	Host candidate
Symmetric	Relay candidate
Permissive	Server reflexive or peer reflexive candidate discovered via STUN (see RFC 5389 RFC 5389: Session Traversal Utilities for NAT (STUN) <http://tools.ietf.org/html/rfc5389>.)

If the client is aware of which type of candidate it is sending, the candidate element MAY contain a 'type' attribute. Although this information is merely a hint about the candidate type, this information can help the recipient be aware of how permissive the peer's NAT or firewall is. The values of the 'type' attribute are "host", "prflx", "relay", and "srflx" (as in the ICE specification and XEP-0176).

4.3 Responder Response

As described in XEP-0166, to acknowledge the session initiation request, the responder returns an IQ-result:

Listing 2: Responder acknowledges the session-initiate request

```
<iq from='juliet@capulet.lit/balcony'
    id='tp2hd816'
    to='romeo@montague.lit/orchard'
    type='result'/>
```

Depending on the application type, a user agent controlled by a human user might need to wait for the user to affirm a desire to proceed with the session before continuing. When the user agent has received such affirmation (or if the user agent can automatically proceed for any reason, e.g. because no human intervention is expected or because a human user has configured the user agent to automatically accept sessions with a given entity), it returns a Jingle session-accept message. This message MUST contain a <transport/> element qualified by the 'urn:xmpp:jingle:transports:raw-udp:1' namespace, which SHOULD in turn contain one <candidate/> element for each Raw UDP candidate generated by or known to the responder.

Listing 3: Responder definitively accepts the session

```
<iq from='juliet@capulet.lit/balcony'
    id='hs81w639'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          initiator='romeo@montague.lit/orchard'
```

```
          responder='juliet@capulet.lit/balcony'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='voice'>
      <description xmlns='urn:xmpp:jingle:apps:rtp:1' media='audio'>
        <payload-type id='18' name='G729'/>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:raw-udp:1'>
        <candidate component='1'
                   generation='0'
                   id='z7sdjb01hf'
                   ip='208.68.163.214'
                   port='9876'/>
        <candidate component='2'
                   generation='0'
                   id='hg92lsn10b'
                   ip='208.68.163.214'
                   port='9877'/>
      </transport>
    </content>
  </jingle>
</iq>
```

(Notice that the foregoing example includes two <candidate/> elements, one for RTP and the other for RTCP.)
The initiator then acknowledges the session acceptance.

Listing 4: Initiator acknowledges session acceptance

```
<iq from='romeo@montague.lit/orchard'
    id='hs81w639'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

4.4 Sending Media

Upon sending the session-accept action, the responder MUST immediately attempt to send media to the initiator. Upon receiving the session-accept action, the initiator MUST immediately attempt to send media to the responder. The exact media to send depends on the application type being negotiated and therefore is out of scope for this specification (e.g., for Jingle RTP Sessions it would be appropriate to send comfort noise as specified in RFC 3389 [7]).
An implementation SHOULD enforce a timeout on receipt of media, such that if no media is received from the other party within a reasonable period of time, the implementation will consider the session to have failed and therefore send to the other party a Jingle "session-terminate" action with a reason code of <timeout/>.

[7] RFC 3389: Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN) <http://tools.ietf.org/html/rfc3389>.

Listing 5: Responder terminates the session

```
<iq from='juliet@capulet.lit/balcony'
    id='rig16s95'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          initiator='romeo@montague.lit/orchard'
          sid='a73sjjvkla37jfea'>
    <reason><timeout/></reason>
  </jingle>
</iq>
```

The other party then acknowledges termination of the session.

Listing 6: Initiator acknowledges termination

```
<iq from='romeo@montague.lit/orchard'
    id='rig16s95'
    to='juliet@capulet.lit/balcony'
    type='result'/>
```

5 Determining Support

To advertise its support for the Jingle Raw UDP Transport Method, when replying to Service Discovery (XEP-0030) [8] information requests an entity MUST return URNs for any version of this protocol that the entity supports -- e.g., "urn:xmpp:jingle:transports:raw-udp:1" for this version and "urn:xmpp:jingle:transports:raw-udp:0" for the previous version (see Namespace Versioning regarding the possibility of incrementing the version number).

Listing 7: Service discovery information request

```
<iq from='romeo@montague.lit/orchard'
    id='uw72g176'
    to='juliet@capulet.lit/balcony'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 8: Service discovery information response

```
<iq from='juliet@capulet.lit/balcony'
    id='uw72g176'
    to='romeo@montague.lit/orchard'
    type='result'>
```

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

```
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='urn:xmpp:jingle:1'/>
    <feature var='urn:xmpp:jingle:transports:raw-udp:0'/>
    <feature var='urn:xmpp:jingle:transports:raw-udp:1'/>
    <feature var='urn:xmpp:jingle:apps:rtp:1'/>
    <feature var='urn:xmpp:jingle:apps:rtp:audio'/>
    <feature var='urn:xmpp:jingle:apps:rtp:video'/>
  </query>
</iq>
```

In order for an application to determine whether an entity supports this protocol, where possible it SHOULD use the dynamic, presence-based profile of service discovery defined in Entity Capabilities (XEP-0115) [9]. However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

6 Security Considerations

6.1 Encryption of Media

A Jingle implementation SHOULD support security preconditions that are enforced before application media is allowed to flow over a UDP association, such as those described in Jingle XTLS [10].
Application types that use the Jingle Raw UDP transport method MAY also define their own application-specific encryption methods, such as the Secure Real-time Transport Protocol (SRTP) for RTP exchanges as described in XEP-0167: Jingle RTP Sessions.

7 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [11].

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

[10] Extensible Messaging and Presence Protocol (XMPP) End-to-End Encryption Using Transport Layer Security ("XTLS") <http://tools.ietf.org/html/draft-meyer-xmpp-e2e-encryption>.

[11] 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/>.

8 XMPP Registrar Considerations

8.1 Protocol Namespaces

This specification defines the following XML namespace:

- urn:xmpp:jingle:transports:raw-udp:1

The XMPP Registrar [12] includes the foregoing namespace in its registry at <https://xmpp.org/registrar/namespaces.html>, as governed by XMPP Registrar Function (XEP-0053) [13].

8.2 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.

8.3 Jingle Transport Methods

The XMPP Registrar includes "raw-udp" in its registry of Jingle transport methods at <https://xmpp.org/registrar/jingle-transports.html>. The registry submission is as follows:

```
<transport>
  <name>raw-udp</name>
  <desc>A method for exchanging data over raw UDP associations.</desc>
  <type>datagram</type>
  <doc>XEP-0177</doc>
</transport>
```

9 XML Schema

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
```

[12] 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/>.

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

```
  targetNamespace='urn:xmpp:jingle:transports:raw-udp:1'
  xmlns='urn:xmpp:jingle:transports:raw-udp:1'
  elementFormDefault='qualified'>

<xs:annotation>
  <xs:documentation>
    The protocol documented by this schema is defined in
    XEP-0177: http://www.xmpp.org/extensions/xep-0177.html
  </xs:documentation>
</xs:annotation>

<xs:element name='transport'>
  <xs:complexType>
    <xs:sequence>
      <xs:element name='candidate'
                  type='candidateElementType'
                  minOccurs='0'
                  maxOccurs='unbounded'/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:complexType name='candidateElementType'>
  <xs:simpleContent>
    <xs:extension base='empty'>
      <xs:attribute name='component' type='xs:unsignedByte' use='
          required'/>
      <xs:attribute name='generation' type='xs:unsignedByte' use='
          required'/>
      <xs:attribute name='id' type='xs:NCName' use='required'/>
      <xs:attribute name='ip' type='xs:string' use='required'/>
      <xs:attribute name='port' type='xs:unsignedShort' use='
          required'/>
      <xs:attribute name='type' use='required'>
        <xs:simpleType>
          <xs:restriction base='xs:NCName'>
            <xs:enumeration value='host'/>
            <xs:enumeration value='prflx'/>
            <xs:enumeration value='relay'/>
            <xs:enumeration value='srflx'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:simpleType name='empty'>
  <xs:restriction base='xs:string'>
```

```
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

10 Acknowledgements

Thanks to Thiago Camargo, Paul Chitescu, Diana Cionoiu, Olivier Crête, Steffen Larsen, Robert McQueen, Mike Ruprecht, Jakob Schroeter, Justin Uberti, Unnikrishnan Vikrama Panicker, and Paul Witty for their feedback.