XEP-0234: Jingle File Transfer

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

Lance Stout
mailto:lance@andyet.com
xmpp:lance@lance.im

2017-05-20
Version 0.18.1

Status	Type	Short Name
Experimental	Standards Track	NOT_YET_ASSIGNED

This specification defines a Jingle application type for transferring a file from one entity to another. The protocol provides a modular framework that enables the exchange of information about the file to be transferred as well as the negotiation of parameters such as the transport to be used.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Jingle (XEP-0166) [1] can be used to initiate and negotiate a wide range of peer-to-peer sessions. One session type of interest is file transfer. This document specifies an application format for negotiating Jingle file transfer sessions, where files are exchanged via any available reliable transport.

2 Requirements

SI File Transfer (XEP-0096) [2] was the original XMPP protocol extension for file transfer negotiation. However, that protocol has several drawbacks, most related to the Stream Initiation (XEP-0095) [3] protocol on which it depends:

1. It does not enable a true, bidirectional negotiation; instead, the initiator sets the terms for the file transfer and the responder either accepts the terms or cancels the negotiation.

2. It is the only technology in the Jabber/XMPP protocol "stack" that uses XEP-0095: Stream Initiation. More modern technologies such as voice and video session negotiation use Jingle (XEP-0166) [4], and it would be helpful if implementors could re-use the same code for all negotiation use cases.

To overcome these drawbacks, this specification defines a file transfer negotiation method that meets the following requirements:

- Use the session negotiation semantics from XEP-0166.

- Use any reliable Jingle transport mechanism, including but not limited to:

  - ICE-TCP RFC 6544 [5]
  - SOCKS5 Bytestreams Jingle SOCKS5 Bytestreams Transport Method (XEP-0260) [6]
  - In-Band Bytestreams Jingle In-Band Bytestreams Transport Method (XEP-0261) [7]

[1] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.
[2] XEP-0096: SI File Transfer <https://xmpp.org/extensions/xep-0096.html>.
[3] XEP-0095: Stream Initiation <https://xmpp.org/extensions/xep-0095.html>.
[4] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.
[5] RFC 6544: TCP Candidates with Interactive Connectivity Establishment (ICE) <http://tools.ietf.org/html/rfc6544>.
[6] XEP-0260: Jingle SOCKS5 Bytestreams Transport Method <https://xmpp.org/extensions/xep-0260.html>.
[7] XEP-0261: Jingle In-Band Bytestreams Transport Method <https://xmpp.org/extensions/xep-0261.html>.

- Define a file description format that, unlike XEP-0096, enables hash agility (via Use of Cryptographic Hash Functions in XMPP (XEP-0300) [8]).

- Define a clear upgrade path from SI File Transfer to Jingle File Transfer.

Note that Jingle file transfer is only as reliable as the transports on which it depends. In particular, SOCKS5 Bytestreams ("S5B") does not always result in NAT or firewall traversal. To work around that problem, this specification requires all implementations to support as a fallback mechanism In-Band Bytestreams ("IBB"), which usually results in a successful (if slow) file transfer. A more robust and adaptable option is ICE-TCP (RFC 6544); at the time of writing Jingle ICE-UDP Transport Method (XEP-0176) [9] is being updated to include the ability to negotiate ICE-TCP candidates.

3 Terminology

File Offer A Jingle File Transfer Content is said to be a File Offer if the content creator is the same as the content sender (see Use of Jingle Content Senders).

File Request A Jingle File Transfer Content is said to be a File Request if the content creator is the opposite of the content sender (see Use of Jingle Content Senders).

File Sender The File Sender is the side of the Jingle session responsible for sending the file data. The File Sender is not necessarily the same entity as the Jingle session initiator, and an entity could be both a File Sender and File Receiver in the context of a single Jingle session with multiple files.

File Receiver The File Receiver is the side of the Jingle session responsible for receiving the file data. The File Receiver is not necessarily the same entity as the Jingle session responder, and an entity could be both a File Receiver and File Sender in the context of a single Jingle session with multiple files.

4 Jingle Conformance

In accordance with Section 12 of XEP-0166, this document specifies the following information related to the Jingle File Transfer ("Jingle FT") application type:

1. The application format negotiation process is defined in the Negotiating a Jingle File Transfer Session section of this document.

[8] XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>.
[9] XEP-0176: Jingle ICE-UDP Transport Method <https://xmpp.org/extensions/xep-0176.html>.

2. The semantics of the <description/> element are defined in the Application Format section of this document.

3. A mapping of Jingle semantics to the Session Description Protocol is provided in the Mapping to Session Description Protocol section of this document.

4. A Jingle File Transfer session SHOULD use a streaming transport method, not a datagram transport method.

5. Transport components are not used in Jingle File Transfer.

6. Content is to be sent and received as follows:
   For streaming transports, outbound content shall be encoded into packets (as defined by the transport mechanism) without any other framing mechanism and sent in succession over the transport. Incoming data received over the transport shall be processed as a stream of packets, where each packet's content payload is entirely composed of the next portion of file data to be processed.

4.1 Use of Jingle Content Senders

Jingle File Transfer makes critical use of the 'senders' attribute of Jingle <content/> elements in order to specify which party is responsible for sending the described file. As such, Jingle File Transfer content MUST include a 'senders' attribute, where the allowed values are "initiator" and "responder". The semantics of the values "both" and "none" are undefined in Jingle File Transfer and thus NOT RECOMMENDED for use with Jingle File Transfer content.
In general, a Jingle File Transfer content is said to be a "File Offer" if the 'senders' attribute is the same as the role of the party adding the content to the session, and a "File Request" if the 'senders' value is the opposite role of the party adding the content.
Note: The content 'creator' attribute does *not* specify who created or is sending the file, it only specifies which party to the session added the Jingle content to the session.

Jingle Session Role	Content Senders	File Transfer Type
initiator	initiator	File Offer
initiator	responder	File Request
responder	initiator	File Request
responder	responder	File Offer

5 Application Format

A Jingle File Transfer session is described by a content type that contains one application format and one transport method. Each <content/> element defines the details of a single file transfer. A Jingle negotiation MAY result in the establishment of multiple file transfers by including multiple <content/> elements.
The application format consists of a file description contained within a <description/> element qualified by the "urn:xmpp:jingle:apps:file-transfer:5" namespace (see Namespace Versioning regarding the possibility of incrementing the version number). The file description is a <file/> element specifying metadata such as the name of the file, media type, etc., as illustrated in the following example.

```
<description xmlns='urn:xmpp:jingle:apps:file-transfer:5'>
  <file>
    <media-type>text/plain</media-type>
    <name>test.txt</name>
    <date>2015-07-26T21:46:00</date>
    <size>6144</size>
    <hash xmlns='urn:xmpp:hashes:2'
          algo='sha-1'>w0mcJylzCn+AfvuGdqkty2+KP48=</hash>
  </file>
</description>
```

The <description/> element is intended to be a child of a Jingle <content/> element as specified in XEP-0166.
The child elements of the <file/> element are as follows:

Element Name	Description	Inclusion
date	UTC timestamp specifying the last modified time of the file (which MUST conform to the DateTime profile of XMPP Date and Time Profiles (XEP-0082) XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.).	OPTIONAL
desc	A human readable description of the file. Multiple <desc/> elements MAY be included if different xml:lang values are specified.	OPTIONAL

Element Name	Description	Inclusion
hash	A hash of the file content, using the <hash/> element defined in Use of Cryptographic Hash Functions in XMPP (XEP-0300) XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>. and qualifed by the 'urn:xmpp:hashes:2' namespace. Multiple hashes MAY be included for hash agility.	REQUIRED when offering a file, otherwise OPTIONAL
media-type	The media type of the file content, which SHOULD be a valid MIME-TYPE as registered with the Internet Assigned Numbers Authority (IANA) 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/>. (specifically, as listed at <http://www.iana.org/assignments/media-types>). If not specified, the content is assumed to be "application/octet-stream".	RECOMMENDED when offering a file, otherwise OPTIONAL
name	The name of the file. The name SHOULD NOT contain characters or character sequences that would be interpreted as a directory structure by the local file system (e.g. "/", "\", "../", etc.). If any such characters or character sequences are present (possibly because the local and remote file systems use different syntax for directory structure), they SHOULD be escaped (e.g., via percent-encoding) before using the name as part of any file system operation. See Security Considerations.	OPTIONAL

Element Name	Description	Inclusion
size	The length of the file's content, in bytes.	OPTIONAL, but SHOULD be present when offering a file.
range	The presence of the <range/> element indicates support of ranged transfers, and can be used to control where a transfer starts.	OPTIONAL

One or more <hash/> elements MUST be present when offering a file, but those elements MAY be empty if the hash has not yet been computed. If there is no computed hash value, the <hash/> element(s) MUST possess an 'algo' attribute specifying which hash algorithm will be used. Once a hash has been calculated by the File Sender, the File Sender SHOULD inform the File Receiver of the hash value as described in Checksum.
Additional elements MAY be included as children of the <file/> element to provide additional metadata about the file, such as File Transfer Thumbnails (XEP-0264) [10].
The optional <range/> element MAY possess two attributes:

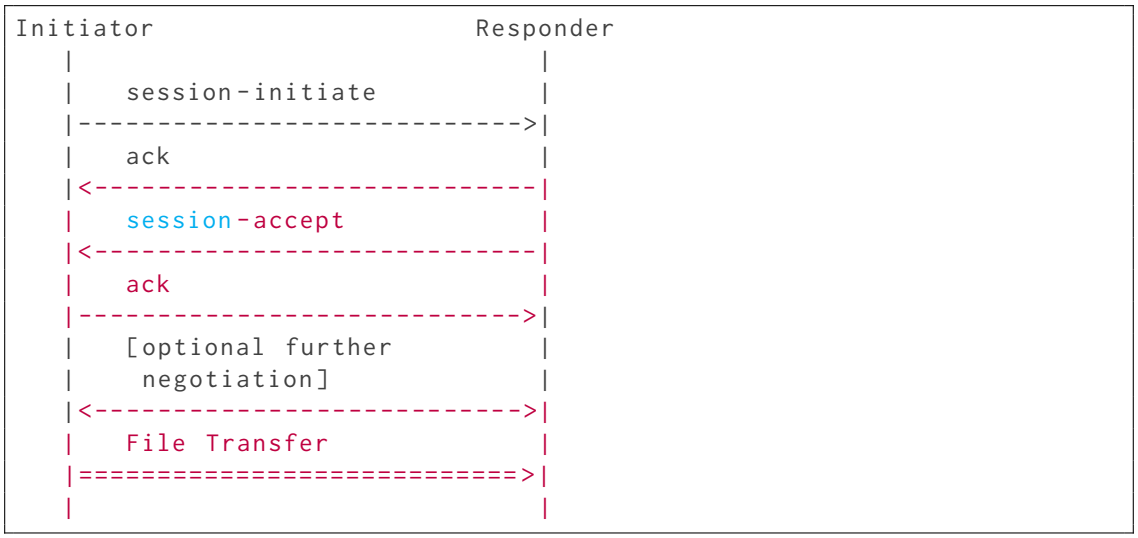
Attribute	Description	Inclusion
offset	Specifies the position, in bytes, from which to start transferring file data. This defaults to zero (0) if not specified.	OPTIONAL
length	Specifies the number of bytes to retrieve starting at offset. This defaults to the length of the file from offset to the end.	OPTIONAL

Inclusion of a <range/> element in a File Offer indicates support of ranged transfers for future File Requests if the transfer is interrupted and needs to be restarted.
A <range/> element MAY include an 'offset' attribute set to begin the transfer at a point other than the start of the file, and MAY include a 'length' attribute to request a portion of the file smaller than the remaining length of the file. If no 'offset' or 'length' attributes are present then it is the same as if no <range/> element was present, because the default values of the attributes would indicate a requested range of the entire file. In general, the first byte of data to be transferred is at the (zero-indexed) position specified by the 'offset' value, with a total of 'length' bytes sent.

[10] XEP-0264: File Transfer Thumbnails <https://xmpp.org/extensions/xep-0264.html>.

6 Negotiating a Jingle File Transfer Session

In general, the process for negotiating a Jingle File Transfer session is as follows:



6.1 Offering a File

To start a File Offer, the initiator sends a Jingle session-initiation request to a potential responder. The request specifies three things:

1. A content 'senders' attribute with the value of 'initiator' to indicate this is a File Offer.
2. An application type of "urn:xmpp:jingle:apps:file-transfer:5". In particular, the <description/> element contains a <file/> elements describing the file to be sent.
3. An appropriate transport method.

In this example, the initiator is <romeo@montague.example>, the responder is <juliet@capulet.example>, the application type is a File Offer, and the transport method is jingle-s5b (XEP-0260).
The flow is as follows.
First the initiator sends a Jingle session-initiate.

Listing 1: Initiator sends session-initiate

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='nzu25s8'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
```

```
          action='session-initiate'
          initiator='romeo@montague.example/dr4hcr0st3lup4c'
          sid='851ba2'>
    <content creator='initiator' name='a-file-offer' senders='
       initiator'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:5'>
        <file>
          <date>1969-07-21T02:56:15Z</date>
          <desc>This is a test. If this were a real file...</desc>
          <media-type>text/plain</media-type>
          <name>test.txt</name>
          <range/>
          <size>6144</size>
          <hash xmlns='urn:xmpp:hashes:2'
                algo='sha-1'>w0mcJylzCn+AfvuGdqkty2+KP48=</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:s5b:1'
                 mode='tcp'
                 sid='vj3hs98y'>
        <candidate cid='hft54dqy'
                   host='192.168.4.1'
                   jid='romeo@montague.example/dr4hcr0st3lup4c'
                   port='5086'
                   priority='8257636'
                   type='direct'/>
        <candidate cid='hutr46fe'
                   host='24.24.24.1'
                   jid='romeo@montague.example/dr4hcr0st3lup4c'
                   port='5087'
                   priority='8258636'
                   type='direct'/>
      </transport>
    </content>
  </jingle>
</iq>
```

Note: Inclusion of the <range/> child of the <file/> element indicates that the initiator supports ranged transfers as described below under Ranged Transfers.

Note: Computing the hash of the file before sending it can slow down the process of file transfer, because the sending application needs to process the file twice. The File Sender might prefer to send the hash after the file transfer has begun, using a session-info message as described under Checksum.

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

Listing 2: Responder acknowledges session-initiate

```
<iq from='juliet@capulet.example/yn0cl4bnw0yr3vym'
    id='nzu25s8'
```

```
    to='romeo@montague.example/dr4hcr0st3lup4c'
    type='result'/>
```

The initiator then attempts to initiate a SOCKS5 Bytestream with the responder as described in XEP-0260 and XEP-0065. In the meantime, the responder returns a Jingle session-accept. In the session-accept message, the <file/> element MAY contain a <range/> element to indicate that the receiver also supports ranged transfers as described below under Ranged Transfers. If the responder includes a <range /> element with a limit or offset, the File Sender SHOULD respect the provided range settings.

Listing 3: Responder sends session-accept

```
<iq from='juliet@capulet.example/yn0cl4bnw0yr3vym'
    id='jn2vs71g'
    to='romeo@montague.example/dr4hcr0st3lup4c'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          responder='juliet@capulet.example/yn0cl4bnw0yr3vym'
          sid='851ba2'>
    <content creator='initiator' name='a-file-offer' senders='
        initiator'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:5'>
        <file>
          <date>1969-07-21T02:56:15Z</date>
          <desc>This is a test. If this were a real file...</desc>
          <media-type>text/plain</media-type>
          <name>test.txt</name>
          <range/>
          <size>6144</size>
          <hash xmlns='urn:xmpp:hashes:2'
                algo='sha-1'>w0mcJylzCn+AfvuGdqkty2+KP48=</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:s5b:1'
                 mode='tcp'
                 sid='vj3hs98y'>
        <candidate cid='ht567dq'
                   host='192.169.1.10'
                   jid='juliet@capulet.example/yn0cl4bnw0yr3vym'
                   port='6539'
                   priority='8257636'
                   type='direct'/>
        <candidate cid='hr65dqyd'
                   host='134.102.201.180'
                   jid='juliet@capulet.example/yn0cl4bnw0yr3vym'
                   port='16453'
                   priority='7929856'
                   type='assisted'/>
```

```
        <candidate cid='grt654q2'
                   host='2001:638:708:30c9:219:d1ff:fea4:a17d'
                   jid='juliet@capulet.example/yn0cl4bnw0yr3vym'
                   port='6539'
                   priority='8257606'
                   type='direct'/>
      </transport>
    </content>
  </jingle>
</iq>
```

The initiator acknowledges the Jingle session-accept.

Listing 4: Initiator acknowledges session-accept

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='jn2vs71g'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='result'/>
```

6.2 Requesting a File

If the File Sender has advertised the existence of a file that it hosts, such as by Publishing Available Jingle Sessions (XEP-0358) [11], or if a previous file transfer attempt has failed and the File Receiver would like to initiate another attempt, the File Receiver can "pull" the file from the File Sender. This is done by sending a Jingle session-initiate to the File Sender which includes a <content/> with the 'senders' attribute set to the opposite Jingle session role of the party requesting the file (see Use of Jingle Content Senders) and a <description/> element qualified by the 'urn:xmpp:jingle:apps:file-transfer:5' namespace and which includes a <file/> element with enough information included to form a "file selector" (see Section 5 of RFC 5547 [12]) to identify the requested file.

Listing 5: File Receiver requests hosted file

```
<iq from='juliet@capulet.example/yn0cl4bnw0yr3vym'
    id='wsn361c3'
    to='romeo@montague.example/dr4hcr0st3lup4c'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='juliet@capulet.example/yn0cl4bnw0yr3vym'
          sid='uj3b2'>
    <content creator='initiator' name='a-file-request' senders='
        responder'>
```

[11] XEP-0358: Publishing Available Jingle Sessions <https://xmpp.org/extensions/xep-0358.html>.
[12] RFC 5547: A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer <http://tools.ietf.org/html/rfc5547>.

```
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:5'>
        <file>
          <hash xmlns='urn:xmpp:hashes:2'
                algo='sha-1'>w0mcJylzCn+AfvuGdqkty2+KP48=</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:s5b:1'
                 mode='tcp'
                 sid='xig361fj'>
        <candidate cid='ht567dq'
                   host='192.169.1.10'
                   jid='juliet@capulet.example/yn0cl4bnw0yr3vym'
                   port='6539'
                   priority='8257636'
                   type='direct'/>
        <candidate cid='hr65dqyd'
                   host='134.102.201.180'
                   jid='juliet@capulet.example/yn0cl4bnw0yr3vym'
                   port='16453'
                   priority='7929856'
                   type='assisted'/>
        <candidate cid='grt654q2'
                   host='2001:638:708:30c9:219:d1ff:fea4:a17d'
                   jid='juliet@capulet.example/yn0cl4bnw0yr3vym'
                   port='6539'
                   priority='8257606'
                   type='direct'/>
      </transport>
    </content>
  </jingle>
</iq>
```

See File not Available for how to respond if the requester does not have permission to request the file, or if the file cannot be found.

6.3 Offering or Requesting Additional Files

While the Jingle File Transfer session is active, either party MAY choose to add additional files (both offers and requests) to the transfer session. To do so, a Jingle content-add action is used, as shown in the following examples.

Listing 6: File Sender offers additional file

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='wsn361c9'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='set'>
```

```
  <jingle xmlns='urn:xmpp:jingle:1'
          action='content-add'
          sid='uj3b2'>
    <content creator='initiator' name='additional' senders='initiator'
        >
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:5'>
        <file>
          <name>second-file.txt</name>
          <media-type>text/plain</media-type>
          <size>6144</size>
          <hash xmlns='urn:xmpp:hashes:2'
                algo='sha-1'>w0mcJylzCn+AfvuGdqkty2+KP48=</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:s5b:1'
                 mode='tcp'
                 sid='vj3hs98y'>
        <candidate cid='hft54dqy'
                   host='192.168.4.1'
                   jid='romeo@montague.example/dr4hcr0st3lup4c'
                   port='5086'
                   priority='8257636'
                   type='direct'/>
        <candidate cid='hutr46fe'
                   host='24.24.24.1'
                   jid='romeo@montague.example/dr4hcr0st3lup4c'
                   port='5087'
                   priority='8258636'
                   type='direct'/>
      </transport>
    </content>
  </jingle>
</iq>
```

The other party then acks the content-add request.

Listing 7: File Receiver acks content-add request

```
<iq from='juliet@capulet.example/yn0cl4bnw0yr3vym'
    to='romeo@montague.example/dr4hcr0st3lup4c'
    id='wsn361c9'
    type='result' />
```

At this point, the content-add request needs to be either accepted or rejected using Jingle content-accept or content-reject actions.

6.4 Ranged Transfers

As in XEP-0096, a transfer can include only part of a file (e.g., to restart delivery of a truncated transfer session at a point other than the start of the file). This is done using the <range/> element. The usage is illustrated in the following examples.
Let us imagine that the parties negotiate a file transfer session using, say, In-Band Bytestreams. During the transfer, the recipient goes offline unexpectedly and IBB stanzas from the File Sender to the File Receiver begin to bounce. When the recipient comes back online, the File Sender could initiate a new Jingle session and specify that it wants to send all chunks after byte 270336 (which might be the 66th chunk of size 4096).

Listing 8: File Sender requests session to send file, with range

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='wsn361c3'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.example/dr4hcr0st3lup4c'
          sid='uj3b2'>
    <content creator='initiator' name='restart' senders='initiator'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:5'>
        <file>
          <range offset='270336'/>
          <hash xmlns='urn:xmpp:hashes:2'
                algo='sha-1'>w0mcJylzCn+AfvuGdqkty2+KP48=</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:s5b:1'
             mode='tcp'
                 sid='vj3hs98y'>
        <candidate cid='hft54dqy'
                   host='192.168.4.1'
                   jid='romeo@montague.example/dr4hcr0st3lup4c'
                   port='5086'
                   priority='8257636'
                   type='direct'/>
        <candidate cid='hutr46fe'
                   host='24.24.24.1'
                   jid='romeo@montague.example/dr4hcr0st3lup4c'
                   port='5087'
                   priority='8258636'
                   type='direct'/>
      </transport>
    </content>
  </jingle>
</iq>
```

6.5 Aborting a Transfer

At any point, either party MAY choose to abort the transfer of a single file, or end the session entirely to abort all active transfers.
When there is only a single Jingle content or if a party wishes to abort the transfer of all files in the session, a session-terminate including a Jingle reason of <cancel /> is sent.

Listing 9: File Receiver aborts all active file transfers

```
<iq from='juliet@capulet.example/yn0cl4bnw0yr3vym'
    id='jp2ba614'
    to='romeo@montague.example/dr4hcr0st3lup4c'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
    <reason>
      <cancel />
    </reason>
  </jingle>
</iq>
```

If a party chooses to abort the transfer of a single file out of several active transfers, a Jingle content-remove action is used, which MAY include a Jingle reason of <cancel/>, as shown in the following example.

Listing 10: File Receiver aborts the transfer

```
<iq from='juliet@capulet.example/yn0cl4bnw0yr3vym'
    id='jp2ba614'
    to='romeo@montague.example/dr4hcr0st3lup4c'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='content-remove'
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='a-file-offer' />
    <reason>
      <cancel />
    </reason>
  </jingle>
</iq>
```

The other party then acks the content-remove request.

Listing 11: File Sender acks abort request

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='jp2ba614'
```

```
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='result' />
```

If after removing the content there are no other Jingle contents the session MUST be terminated as described in the next section.

6.6 Ending the Session

Once all file content in the session has been transfered, either party MAY acknowledge receipt of the received files (see Received) or, if there are no other active file transfers, terminate the Jingle session with a Jingle session of <success/>. Preferably, sending the session-terminate is done by the last entity to finish receiving a file to ensure that all offered or requested files by either party have been completely received (up to the advertised sizes).

Listing 12: File Receiver ends the session after successfully receiving all files

```
<iq from='juliet@capulet.example/yn0cl4bnw0yr3vym'
    id='jp2ba614'
    to='romeo@montague.example/dr4hcr0st3lup4c'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-terminate'
          sid='a73sjjvkla37jfea'>
    <reason>
      <success />
    </reason>
  </jingle>
</iq>
```

7 Mapping to Session Description Protocol

RFC 5547 [13] defines the general process for including file transfer information in SDP.
The SDP media type for Jingle File Transfer can be "message" (e.g. when used with RFC 4975 [14]) or "application"; however, this media value is not reflected in the Jingle File Transfer application format.
Any combination of <name/>, <size/>, <media-type/>and <hash/> values MAY be used to form a "file selector" (see Section 5 of RFC 5547 [15]), which would be mapped to SDP as follows:

[13] RFC 5547: A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer <http://tools.ietf.org/html/rfc5547>.
[14] RFC 4975: The Message Session Relay Protocol (MSRP) <http://tools.ietf.org/html/rfc4975>.
[15] RFC 5547: A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer <http://tools.ietf.org/html/rfc5547>.

```
a=file-selector [name:"<name>"] [size:<size>] [type:<media-type>] [
    hash:<hash algo>:<hash value>]
```

(The hash value MUST be encoded as hexadecimal with each byte separated by a colon.)
The <date/> value is the last modified time of the file, and thus is mapped as follows:

```
a=file-date:modification:"<date>"
```

Note: the format used here for <date> is the date-time format defined in RFC 5322 [16].
If a range is specified, the SDP mapping requires both a start and stop offset. If no length was specified for the range, the stop offset is "*". If a length was specified, the stop offset is the <range/> offset value plus the length.

```
a=file-range:<offset>-<(offset + length) | *>
```

As a full example, given the following Jingle File Transfer content description:

```
<description xmlns='urn:xmpp:jingle:apps:file-transfer:5'>
  <file>
    <media-type>text/plain</media-type>
    <name>test.txt</name>
    <date>2015-07-26T21:46:00</date>
    <size>6144</size>
    <hash xmlns='urn:xmpp:hashes:2'
          algo='sha-1'>w0mcJylzCn+AfvuGdqkty2+KP48=</hash>
    <range offset='1024' />
  </file>
</description>
```

The equivalent SDP would be:

```
a=file-selector: name:"test.txt" size:6144 type:text/plain hash:sha-1
    :55:2D:A7:49:93:08:52:C6:9A:E5:D2:14:1D:37:66:B1
a=file-date:modification:"26 Jul 2015 21:46:00"
a=file-range:1024-*
```

8 Informational Messages

8.1 Received

Once a file has been successfully received, the recipient MAY send a Jingle session-info message indicating receipt of the complete file, which consists of a <received/> element

[16] RFC 5322: Internet Message Format <http://tools.ietf.org/html/rfc5322>.

qualified by the 'urn:xmpp:jingle:apps:file-transfer:5' namespace. The <received/> element SHOULD contain 'creator' and 'name' attributes sufficient to identify the content that was received.

Listing 13: File Receiver sends ack in session-info

```
<iq from='juliet@capulet.example/yn0cl4bnw0yr3vym'
    id='jp2ba614'
    to='romeo@montague.example/dr4hcr0st3lup4c'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          sid='a73sjjvkla37jfea'>
    <received xmlns='urn:xmpp:jingle:apps:file-transfer:5'
              creator='initiator'
              name='a-file-offer' />
  </jingle>
</iq>
```

8.2 Checksum

At any time during the lifetime of the file transfer session, the File Sender can communicate the checksum of the file to the File Receiver.
This can be done in the session-initiate message if the File Sender already knows the checksum, as shown above in Example 3.
After the session-initiate message, this can also be done by sending a session-info message containing a <checksum/> element qualified by the 'urn:xmpp:jingle:apps:file-transfer:5' namespace. The <checksum/> element SHOULD contain 'creator' and 'name' attributes sufficient to identitfy the content the checksum belongs to. Additionally, the <checksum/> element MUST contain a <file/> element which MUST contain at least one <hash/> element qualified by the 'urn:xmpp:hashes:2' namespace. Each <hash/> element contains a checksum of the file data produced in accordance with the hashing function specified by the 'algo' attribute, which MUST be one of the functions listed in the IANA Hash Function Textual Names Registry [17].

Listing 14: Initiator sends checksum in session-info

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='kqh401b5'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
```

[17] IANA registry of Hash Function Textual Names <http://www.iana.org/assignments/hash-function-text-names>.

```
          sid='a73sjjvkla37jfea'>
    <checksum xmlns='urn:xmpp:jingle:apps:file-transfer:5'
              creator='initiator'
              name='a-file-offer'>
      <file>
        <hash xmlns='urn:xmpp:hashes:2'
              algo='sha-1'>w0mcJylzCn+AfvuGdqkty2+KP48=</hash>
      </file>
    </checksum>
  </jingle>
</iq>
```

If a ranged transfer was requested, the <file/> element inside the <checksum/> element MAY include a <range/> element specifying the offset and length of the requested range, which in turn includes <hash/> element(s) with hashes of the data that was transferred for that range.

Listing 15: Initiator sends checksum in session-info for a specific range

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='kqh401b5'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-info'
          sid='a73sjjvkla37jfea'>
    <checksum xmlns='urn:xmpp:jingle:apps:file-transfer:5'
              creator='initiator'
              name='a-file-offer'>
      <file>
        <range offset='2048' length='1024'>
          <hash xmlns='urn:xmpp:hashes:2'
                algo='sha-1'>kHp5RSzW/h7Gm1etSf90Mr5PC/k=</hash>
        </range>
      </file>
    </checksum>
  </jingle>
</iq>
```

If the initiator wishes to communicate only the hashing algorithm at the beginning of the session (e.g., because it has not yet calculated the checksum), it can send an empty <hash/> element (without a checksum in the XML character data as shown in the previous examples) in the session-initiate message; this enables the recipient to check the file during the transfer session (which can be helpful in the case of transfers that are truncated or fail mid-stream).

Listing 16: Initiator communicates hashing algorithm in session-initiate

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='nzu25s8'
```

```
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.example/dr4hcr0st3lup4c'
          sid='851ba2'>
    <content creator='initiator' name='a-file-offer' senders='
        initiator'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:5'>
        <file>
          <date>1969-07-21T02:56:15Z</date>
          <desc>This is a test. If this were a real file...</desc>
          <media-type>text/plain</media-type>
          <name>test.txt</name>
          <range/>
          <size>6144</size>
          <hash xmlns='urn:xmpp:hashes:2' algo='sha-1'/>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:s5b:1'
                 mode='tcp'
                 sid='vj3hs98y'>
        <candidate cid='hft54dqy'
                   host='192.168.4.1'
                   jid='romeo@montague.example/dr4hcr0st3lup4c'
                   port='5086'
                   priority='8257636'
                   type='direct'/>
        <candidate cid='hutr46fe'
                   host='24.24.24.1'
                   jid='romeo@montague.example/dr4hcr0st3lup4c'
                   port='5087'
                   priority='8258636'
                   type='direct'/>
      </transport>
    </content>
  </jingle>
</iq>
```

9 Errors

9.1 File not Available

If a requested file cannot be found (or the requester does not have permission to request or know about the existence of the file in question), then the File Sender SHOULD send either a session-terminate or content-reject action in response to the session-initiate or content-add request, and SHOULD include a Jingle reason of <failed-application/> and MAY

include an application specific reason of a <file-not-available/> element qualified by the 'urn:xmpp:jingle:apps:file-transfer:errors:0' namespace.

Listing 17: File Sender rejects file request because the file could not be found

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='wsn361c9'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='content-reject'
          sid='uj3b2'>
    <content creator='initiator' name='requesting-file' senders='
        initiator'>
    <reason>
      <failed-application />
      <file-not-available xmlns='urn:xmpp:jingle:apps:file-
          transfer:errors:0' />
    </reason>
  </jingle>
</iq>
```

9.2 File too Large

There are several situations where a File Receiver might wish to abort a transfer due to an excess of file data, for example:

- The File Receiver has reached a file system storage quota or other hard limit that prevents continuing to receive file data.
- The File Sender has continued sending data past the initially specified size of the file.

In such cases, the File Receiver MAY abort the transfer by sending a Jingle session-terminate (or content-remove as appropriate) which includes a Jingle reason of <media-error/> and MAY include an application specific reason of a <file-too-large/> element qualified by the 'urn:xmpp:jingle:apps:file-transfer:errors:0' namespace.

Listing 18: File Receiver rejects file offer because the file is too large

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='wsn361c9'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='content-remove'
          sid='uj3b2'>
```

```
    <content creator='initiator' name='big-file' senders='initiator'>
    <reason>
      <media-error />
      <file-too-large xmlns='urn:xmpp:jingle:apps:file-
          transfer:errors:0' />
    </reason>
  </jingle>
</iq>
```

To prevent denial of service and other attacks, the File Receiver is fully within its rights to drop received data or not send a session-terminate message.

10 Implementation Notes

10.1 Mandatory to Implement Technologies

All implementations MUST support the Jingle In-Band Bytestreams Transport Method (XEP-0261) as a reliable method of last resort. An implementation SHOULD support other transport methods as well, especially ICE-TCP (RFC 6544) and the Jingle SOCKS5 Bytestreams Transport Method (XEP-0260).

10.2 Preference Order of Transport Methods

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

10.3 Migration from XEP-0096

Support for Jingle file transfer can be determined through discovery of the 'urn:xmpp:jingle:apps:file-transfer:5' namespace (see Namespace Versioning regarding the possibility of incrementing the version number), via either service discovery (XEP-0030) or entity capabilities (XEP-0115). If the initiator knows that the responder supports Jingle file transfer, it SHOULD first attempt negotiation using Jingle rather than Stream Initiation.

11 Determining Support

To advertise its support for the Jingle File Transfer, when replying to service discovery information ("disco#info") requests an entity MUST return URNs for any version of this protocol that the entity supports -- e.g., "urn:xmpp:jingle:apps:file-transfer:5" for this version (see Namespace Versioning regarding the possibility of incrementing the version number).

Listing 19: Service discovery information request

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='uw72g176'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 20: Service discovery information response

```
<iq from='juliet@capulet.example/yn0cl4bnw0yr3vym'
    id='uw72g176'
    to='romeo@montague.example/dr4hcr0st3lup4c'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='urn:xmpp:jingle:1'/>
    <feature var='urn:xmpp:jingle:apps:file-transfer:5'/>
    <feature var='urn:xmpp:jingle:transports:s5b:1'/>
    <feature var='urn:xmpp:jingle:transports:ibb:1'/>
  </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) [18]. However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

12 Security Considerations

Caution needs to be exercised when using the <name/> of a file offer or request to control any interaction with a file system. For example, a malicious user could request a file with <name>/etc/passwd</name> or include file system specific control patterns such as <name>../../private.txt</name> to try and access a sensitive file outside of the set of files intended to be shared. Or a malicious user could offer a file named "/etc/passwd" to try and trick the receiver into overwriting that or other sensitive files. Therefore, implementations SHOULD escape any file system path separators in the <name/> before using that value in any file system calls.

It is RECOMMENDED for implementations to use the strongest hashing algorithm available to both parties. See XEP-0300 for further discussion.

In order to secure the data stream, implementations SHOULD use encryption methods appropriate to the transport method being used. For example, end-to-end encryption can be negotiated over either SOCKS5 Bytestreams or In-Band Bytestreams as described in XEP-0260 and XEP-0261.

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

Refer to XEP-0047, XEP-0065, XEP-0096, XEP-0176, XEP-0260, XEP-0261, and RFC 6544 for related security considerations.

13 IANA Considerations

No interaction with the Internet Assigned Numbers Authority (IANA) [19] is required as a result of this document.
The XML character data of the <media-type/> element SHOULD be a value registered with the IANA in the IANA MIME Media Types Registry [20].

14 XMPP Registrar Considerations

14.1 Protocol Namespaces

This specification defines the following XML namespace:

- urn:xmpp:jingle:apps:file-transfer:5

Upon advancement of this specification from a status of Experimental to a status of Draft, the XMPP Registrar [21] shall add the foregoing namespace to the registry located at <https://xmpp.org/registrar/namespaces.html>, as described in Section 4 of XMPP Registrar Function (XEP-0053) [22].

14.2 Namespace 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.

[19] 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/>.
[20] IANA registry of MIME media types <http://www.iana.org/assignments/media-types>.
[21] 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/>.
[22] XEP-0053: XMPP Registrar Function <https://xmpp.org/extensions/xep-0053.html>.

14.3 Jingle Application Formats

The XMPP Registrar shall include ”file-transfer” in its registry of Jingle application formats. The registry submission is as follows:

```
<application>
  <name>file-transfer</name>
  <desc>Jingle sessions for the transfer of a file</desc>
  <transport>streaming</transport>
  <doc>XEP-0234</doc>
</application>
```

15 XML Schema

15.1 urn:xmpp:jingle:apps:file-transfer:5

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:jingle:apps:file-transfer:5'
    xmlns='urn:xmpp:jingle:apps:file-transfer:5'
    elementFormDefault='qualified'>

  <xs:import namespace='urn:xmpp:hashes:2'/>

  <xs:element name='description'>
    <xs:complexType>
      <xs:all>
        <xs:element name='file' type='fileTransferElementType'/>
      </xs:all>
    </xs:complexType>
  </xs:element>

  <xs:element name='checksum'>
    <xs:complexType>
      <xs:all>
        <xs:element name='file' type='fileTransferElementType'/>
      </xs:all>
      <xs:attribute name='creator' use='required'>
        <xs:simpleType>
          <xs:restriction base='xs:NCName'>
            <xs:enumeration value='initiator'/>
            <xs:enumeration value='responder'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
```

```
      <xs:attribute name='name' type='xs:string' use='required'/>
    </xs:complexType>
  </xs:element>

  <xs:element name='received'>
    <xs:complexType>
      <xs:attribute name='creator' use='required'>
        <xs:simpleType>
          <xs:restriction base='xs:NCName'>
            <xs:enumeration value='initiator'/>
            <xs:enumeration value='responder'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name='name' type='xs:string' use='required'/>
    </xs:complexType>
  </xs:element>

  <xs:complexType name='fileTransferElementType'>
    <xs:all xmlns:h='urn:xmpp:hashes:2' minOccurs='0'>
      <xs:element name='date' type='xs:date'/>
      <xs:element name='media-type' type='xs:string'/>
      <xs:element name='name' type='xs:string'/>
      <xs:element name='desc' type='xs:string'/>
      <xs:element name='size' type='xs:positiveInteger'/>
      <xs:element name='range' type='fileTransferRangeType'/>
      <xs:element ref='h:hash' minOccurs='0' maxOccurs='unbounded'/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name='fileTransferRangeType'>
    <xs:attribute name='offset' type='xs:nonNegativeInteger' use='
       optional' default='0' />
    <xs:all xmlns:h='urn:xmpp:hashes:2' minOccurs='0'>
      <xs:element ref='h:hash' minOccurs='0' maxOccurs='unbounded' />
    </xs:all>
  </xs:complexType>

</xs:schema>
```

15.2 urn:xmpp:jingle:apps:file-transfer:errors:0

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:jingle:apps:file-transfer:5'
    xmlns='urn:xmpp:jingle:apps:file-transfer:errors:0'
```

```
    elementFormDefault='qualified'>

  <xs:element name='file-not-available' type='empty' />
  <xs:element name='file-too-large' type='empty' />

</xs:schema>
```

16 Acknowledgements

Thanks to Diana Cionoiu, Olivier Crête, Viktor Fast, Philipp Hancke, Waqas Hussain, Justin Karneges, Steffen Larsen, Yann Leboulanger, Marcus Lundblad, Robert McQueen, Joe Maissel, Glenn Maynard, Ali Sabil, Sjoerd Simons, Will Thompson, Matthew Wild, and Jiří Zárevúcky for their feedback.