XEP-0370: Jingle HTTP Transport Method

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

2016-01-12
Version 0.1

Status	Type	Short Name
Experimental	Standards Track	NOT_YET_ASSIGNED

This specification defines two Jingle transport methods for establishing HTTP connections for either uploading or downloading data.

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 two transport methods for establishing and managing data exchanges between XMPP entities using the Hyper Text Transfer Protocol (HTTP, see RFC 2616 [2]); one method is for sharing via pulling data from an HTTP URI (http-download), and the other is for sharing via pushing data to an HTTP URI (http-upload).

2 Requirements

Historically, Out-of-Band Data (XEP-0066) [3] has been used to trigger downloading files via HTTP, as well as initiating the use of any other known URI scheme. However, it has several limitations:

- It only allows for the downloading of files via HTTP, not uploading.
- It can not specify any additional headers that might be necessary for downloading files (authentication tokens, etc).
- The allowance of any URI scheme makes it too unwieldy to be integrated into Jingle the way SOCKS5 Bytestreams and In-Band Bytestreams were (see Jingle SOCKS5 Bytestreams Transport Method (XEP-0260) [4] and Jingle In-Band Bytestreams Transport Method (XEP-0261) [5]).

As such, this document defines two Jingle mechanisms designed to meet the following requirements:

- Allow for both uploading and downloading of data via HTTP.
- Allow additional HTTP headers to be specified.
- Restrict the URI scheme to HTTP/HTTPS.

[1] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.
[2] RFC 2616: Hypertext Transport Protocol -- HTTP/1.1 <http://tools.ietf.org/html/rfc2616>.
[3] XEP-0066: Out of Band Data <https://xmpp.org/extensions/xep-0066.html>.
[4] XEP-0260: Jingle SOCKS5 Bytestreams Transport Method <https://xmpp.org/extensions/xep-0260.html>.
[5] XEP-0261: Jingle In-Band Bytestreams Transport Method <https://xmpp.org/extensions/xep-0261.html>.

3 Jingle Conformance

In accordance with Section 12 of XEP-0166, this document specifies the following information related to both the Jingle http-download and http-upload transport methods:

1. The transport negotiation process for http-download is defined in the Negotiating HTTP Download section of this document, and the negotation process for http-upload is defined in the Negotiating HTTP Upload section of this document.

2. The semantics of the <transport/> element are defined in the Negotiating HTTP Download and Negotiating HTTP Upload sections of this document.

3. Successful negotiation of both the http-download and http-upload methods results in use of a streaming transport method suitable for use in Jingle application types where packet loss cannot be tolerated (e.g., file transfer).

4. Multiple components are not supported by http-download or http-upload.

4 Negotiating HTTP Download

Negotiating HTTP downloads is done by using a <transport/> element with the 'urn:xmpp:jingle:transports:http:0' namespace, (see Namespace Versioning regarding the possibility of incrementing the version number). This element MAY include <candidate/> elements which represent URIs where data can be downloaded. Each <candidate/> element MUST include a 'uri' attribute, and MAY contain <header/> elements whose 'name' attribute is an HTTP header and whose text content is the HTTP header value.

```
<transport xmlns='urn:xmpp:jingle:transports:http:0'>
  <candidate uri='https://files.montague.example/ERUN8970'>
    <header name='authorization'>Bearer 7472327205
        ffb74d10b11363044d8c24e3ddba12</header>
  </candidate>
</transport>
```

Multiple candidates MAY be provided, indicating that there are multiple URIs from which the data can be retrieved (e.g. multiple candidates could be included to list the primary URI of a file along with several known mirrors).

```
<transport xmlns='urn:xmpp:jingle:transports:http:0'>
  <candidate uri='https://files.montague.example/ERUN8970' />
```

```
  <candidate uri='https://mirror1.montague.example/ERUN8970' />
  <candidate uri='https://mirror2.montague.example/ERUN8970' />
</transport>
```

The generation of candidates is based on the Jingle content senders, and only the parties specified to send data SHOULD provide candidates.
Upon receiving an HTTP download candidate, parties that are to receive data (based on the Jingle content senders) SHOULD use an HTTP GET request to the candidate URI to fetch the data.
Entities MAY initially provide an empty set of candidates if a suitable download URI is not yet known; advertising candidates later is done with transport-info actions.

Content Creator	Content Senders	Who Sends Download Candidates	Who performs HTTP GET
initiator	initiator	initiator	responder
responder	responder	initiator	
both	both	both	
none	none	none	
responder	initiator	initiator	responder
responder	responder	initiator	
both	both	both	
none	none	none	

5 Negotiating HTTP Upload

Negotiating HTTP uploads is done by using a <transport/> element with the 'urn:xmpp:jingle:transports:http:upload:0' namespace, (see Namespace Versioning regarding the possibility of incrementing the version number). This element MAY include a <candidate/> element which represents a URI where data can be uploaded. The <candidate/> element MUST include a 'uri' attribute, and MAY contain <header/> elements whose 'name' attribute is an HTTP header and whose text content is the HTTP header value.

```
<transport xmlns='urn:xmpp:jingle:transports:http:upload:0'>
  <candidate uri='https://files.montague.example/ERUN8970'>
    <header name='authorization'>Bearer 7472327205
       ffb74d10b11363044d8c24e3ddba12</header>
  </candidate>
</transport>
```

The generation of candidates is based on the Jingle content senders, and only the parties specified to receive data SHOULD provide candidates.

Upon receiving an HTTP upload candidate, parties that are to send data (based on the Jingle content senders) SHOULD use an HTTP PUT request to the candidate URI, where the request body is the data to be transferred.

Content Creator	Content Senders	Who Sends Upload Candidates	Who Performs HTTP PUT
initiator	initiator	responder	initiator
responder	initiator	responder	
both	both	both	
none	none	none	
responder	initiator	responder	initiator
responder	initiator	responder	
both	both	both	
none	none	none	

See Upload Complete for signaling that the upload process has been completed.

6 Informational Messages

6.1 Upload Complete

A common case for using http-upload is to delegate the storage of the uploaded data to an external hosting service, which means that the receiver might not have the direct ability to know when the uploaded data is ready.
As such, when an upload transfer is used, the party uploading content SHOULD signal when the upload has completed by sending a Jingle transport-info event that specifies the content for which uploading has completed, and includes a <transport/> element qualified by the 'urn:xmpp:jingle:transports:http:upload:0' namespace, which in turn contains a <completed /> element.

Listing 1: Signaling that the upload has completed

```
<iq from='romeo@montague.lit/orchard'
    id='uw72g176'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='transport-info'>
    <content creator='initiator' name='file-upload'>
      <transport xmlns='urn:xmpp:jingle:transports:http:upload:0'>
        <completed />
```

```
      </transport>
    </content>
  </jingle>
</iq>
```

7 Examples

7.1 Offering a File using HTTP Download

Here, Romeo is offering to send a file to Juliet, so he includes a download URI candidate with his session-initiate.

Listing 2: Offering a file with a download URI

```
<iq from='romeo@montague.lit/orchard'
    id='nzu25s8'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.lit/orchard'
          sid='851ba2'>
    <content creator='initiator' name='a-file-offer' senders='
        initiator'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:4'>
        <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:1' algo='sha-1'>552
              da749930852c69ae5d2141d3766b1</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:http:0'>
        <candidate uri='https://files.montague.example/test.txt' />
      </transport>
    </content>
  </jingle>
</iq>
```

Juliet accepts the offer, and then performs an HTTP GET to retrieve the file.

Listing 3: Accepting an offered file using HTTP download

```
<iq from='juliet@capulet.lit/balcony'
    id='nzu25s8'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          initiator='romeo@montague.lit/orchard'
          sid='851ba2'>
    <content creator='initiator' name='a-file-offer' senders='
        initiator'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:4' />
      <transport xmlns='urn:xmpp:jingle:transports:http:0' />
    </content>
  </jingle>
</iq>
```

7.2 Requesting a File using HTTP Download

Here Romeo is requesting Juliet to send a file by sharing a download URI.

Listing 4: Requesting a file and download URI

```
<iq from='romeo@montague.lit/orchard'
    id='nzu25s8'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.lit/orchard'
          sid='851ba2'>
    <content creator='initiator' name='a-file-request' senders='
        responder'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:4'>
        <file>
          <hash xmlns='urn:xmpp:hashes:1' algo='sha-1'>552
              da749930852c69ae5d2141d3766b1</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:http:0' />
    </content>
  </jingle>
</iq>
```

Juliet accepts the request, and includes a download URI in her session-accept.

Listing 5: Returning download URI

```
<iq from='juliet@capulet.lit/balcony'
```

```
    id='nzu25s8'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          sid='851ba2'>
    <content creator='initiator' name='a-file-request' senders='
        responder'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:4' />
        <file>
          <date>1969-07-21T02:56:15Z</date>
          <media-type>text/plain</media-type>
          <name>test.txt</name>
          <range/>
          <size>6144</size>
          <hash xmlns='urn:xmpp:hashes:1' algo='sha-1'>552
              da749930852c69ae5d2141d3766b1</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:http:0'>
        <candidate uri='https://files.capulet.example/test.txt' />
      </transport>
    </content>
  </jingle>
</iq>
```

Romeo then retrieves the file using an HTTP GET request.

7.3 Offering a File using HTTP Upload

In this case, Romeo is offering a file to Juliet but wishes to upload it to her.

Listing 6: Offering to upload a file

```
<iq from='romeo@montague.lit/orchard'
    id='nzu25s8'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.lit/orchard'
          sid='851ba2'>
    <content creator='initiator' name='a-file-offer' senders='
        initiator'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:4'>
        <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:1' algo='sha-1'>552
              da749930852c69ae5d2141d3766b1</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:http:upload:0' />
    </content>
  </jingle>
</iq>
```

Juliet accepts, and provides a candidate with an upload URI that includes an authorization header.

Listing 7: Accepting the session and providing an upload URI

```
<iq from='juliet@capulet.lit/balcony'
    id='nzu25s8'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          sid='851ba2'>
    <content creator='initiator' name='a-file-request' senders='
        responder'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:4' />
      <transport xmlns='urn:xmpp:jingle:transports:http:upload:0'>
        <candidate uri='https://files.capulet.example/ERIE32430'>
          <header name='authorization'>Bearer 7472327205
              ffb74d10b11363044d8c24e3ddba12</header>
        </candidate>
      </transport>
    </content>
  </jingle>
</iq>
```

Romeo now uses an HTTP PUT to upload his file. Once the upload is complete, he informs Juliet so she knows the file is ready for to read.

Listing 8: Signaling that the upload has completed

```
<iq from='romeo@montague.lit/orchard'
    id='uw72g176'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='transport-info'
```

```
          sid='851ba2'>
    <content creator='initiator' name='file-upload'>
      <transport xmlns='urn:xmpp:jingle:transports:http:upload:0'>
        <completed />
      </transport>
    </content>
  </jingle>
</iq>
```

7.4 Requesting a File using HTTP Upload

Here Romeo asks Juliet to upload a file.

Listing 9: Offering a file with a download URI

```
<iq from='romeo@montague.lit/orchard'
    id='nzu25s8'
    to='juliet@capulet.lit/balcony'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.lit/orchard'
          sid='851ba2'>
    <content creator='initiator' name='a-file-offer' senders='
        initiator'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:4'>
        <file>
          <hash xmlns='urn:xmpp:hashes:1' algo='sha-1'>552
              da749930852c69ae5d2141d3766b1</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:http:upload:0'>
        <candidate uri='https://files.montague.example/test.txt' />
      </transport>
    </content>
  </jingle>
</iq>
```

Juliet accepts the session, and begins uploading the file data with an HTTP PUT request.

Listing 10: Accepting the session

```
<iq from='juliet@capulet.lit/balcony'
    id='nzu25s8'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
```

```
          sid='851ba2'>
    <content creator='initiator' name='a-file-request' senders='
        responder'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:4' />
        <file>
          <date>1969-07-21T02:56:15Z</date>
          <media-type>text/plain</media-type>
          <name>test.txt</name>
          <range/>
          <size>6144</size>
          <hash xmlns='urn:xmpp:hashes:1' algo='sha-1'>552
              da749930852c69ae5d2141d3766b1</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:http:upload:0'>
    </content>
  </jingle>
</iq>
```

Once the upload is complete, she informs Romeo that she has completed the upload so that he knows he can access the data he requested.

Listing 11: Signaling that the upload has completed

```
<iq from='juliet@capulet.lit/balcony'
    id='uw72g176'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='transport-info'
          sid='851ba2'>
    <content creator='initiator' name='file-upload'>
      <transport xmlns='urn:xmpp:jingle:transports:http:upload:0'>
        <completed />
      </transport>
    </content>
  </jingle>
</iq>
```

8 Determining Support

To advertise its support for the Jingle HTTP Transport Method, when replying to Service Discovery (XEP-0030) [6] information requests an entity MUST return URNs for any version of this protocol that the entity supports -- e.g., "urn:xmpp:jingle:transports:http:0" for this version (see Namespace Versioning regarding the possibility of incrementing the version

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

number).

Listing 12: 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 13: Service discovery information response

```
<iq from='juliet@capulet.lit/balcony'
    id='uw72g176'
    to='romeo@montague.lit/orchard'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='urn:xmpp:jingle:1'/>
    <feature var='urn:xmpp:jingle:transports:http:0'/>
    <feature var='urn:xmpp:jingle:transports:http:upload:0'/>
  </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) [7]. However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

9 Security Considerations

HTTP URI candidates SHOULD use the "https://" URI scheme instead of "http://", and entities MAY refuse to process URIs that are not "https://".
Certain HTTP headers can cause unintended behaviour, such as using the 'Upgrade' header to trigger a conversion to WebSocket (RFC 6455 [8]).

10 IANA Considerations

No interaction with the Internet Assigned Numbers Authority (IANA) [9] is required as a result of this document.

[7] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[8] RFC 6455: The WebSocket Protocol <http://tools.ietf.org/html/rfc6455>.
[9] 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/>.

11 XMPP Registrar Considerations

11.1 Protocol Namespaces

This specification defines the following XML namespaces:

- urn:xmpp:jingle:transports:http:0
- urn:xmpp:jingle:transports:http:upload:0
- urn:xmpp:jingle:transports:http:info:0

The XMPP Registrar [10] includes the foregoing namespace in its registry of protocol namespaces at <https://xmpp.org/registrar/namespaces.html>, as described in Section 4 of XMPP Registrar Function (XEP-0053) [11].

11.2 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.

11.3 Jingle Transport Methods

The XMPP Registrar [12] includes "http-download" in its registry of Jingle transport methods at <https://xmpp.org/registrar/jingle-transports.html>. The registry submission is as follows:

```
<transport>
  <name>http-download</name>
  <desc>
    A method for negotiating data exchange via HTTP URI retrieval.
  </desc>
```

[10] 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/>.

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

[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/>.

```
  <type>streaming</type>
  <doc>XEP-XXXX</doc>
</transport>
```

The XMPP Registrar [13] includes "http-upload" in its registry of Jingle transport methods at <https://xmpp.org/registrar/jingle-transports.html>. The registry submission is as follows:

```
<transport>
  <name>http-upload</name>
  <desc>
    A method for negotiating data exchange via uploading to HTTP URIs.
  </desc>
  <type>streaming</type>
  <doc>XEP-XXXX</doc>
</transport>
```

12 XML Schema

12.1 urn:xmpp:jingle:transports:http:0

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

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

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

  <xs:complexType name='httpSlotType'>
    <xs:all minOccurs='0'>
      <xs:element ref='header' maxOccurs='unbounded' />
    </xs:all>
    <xs:attribute name='uri' type='xs:anyURI' use='required' />
```

[13] 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/>.

```
  </xs:complexType>

  <xs:element name='header'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:string'>
          <xs:attribute name='name' use='required'/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

12.2 urn:xmpp:jingle:transports:http:upload:0

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

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

  <xs:element name='transport'>
    <xs:complexType>
      <xs:all minOccurs='0'>
        <xs:element name='completed' maxOccurs='1' />
        <xs:element name='candidate' type='httpSlotType' maxOccurs='
            unbounded' />
      </xs:all>
    </xs:complexType>
  </xs:element>

  <xs:complexType name='httpSlotType'>
    <xs:all minOccurs='0'>
      <xs:element ref='header' maxOccurs='unbounded' />
    </xs:all>
    <xs:attribute name='uri' type='xs:anyURI' use='required' />
  </xs:complexType>

  <xs:element name='header'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:string'>
          <xs:attribute name='name' use='required'/>
        </xs:extension>
      </xs:simpleContent>
```

```
    </xs:complexType>
  </xs:element>

</xs:schema>
```