XEP-0363: HTTP File Upload

Daniel Gultsch
mailto:daniel@gultsch.de
xmpp:daniel@gultsch.de

2025-06-17
Version 1.2.0

Status	Type	Short Name
Draft	Standards Track	NOT_YET_ASSIGNED

This specification defines a protocol to request permissions from another entity to upload a file to a specific path on an HTTP server and at the same time receive a URL from which that file can later be downloaded again.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

XMPP protocol extensions already define methods for peer-to-peer file transfer such as SI File Transfer (XEP-0096) [1] or Jingle File Transfer (XEP-0234) [2] however due to their very nature of being peer-to-peer they don't work very well in scenarios where it is requeried to send a file to multiple recipients or multiple resources of the same recipient at once. They also don't work alongside offline storage, MUC history and Message Archive Management (XEP-0313) [3]. Uploading files manually to an HTTP server and sharing the link has been a workaround for this for a long time now. While users have a variety of services to choose from the downside of this manual approach is that an XMPP client can not automate this process on behalf of the user since these services don't share a common API. Furthermore using a third party service would probably require the user to enter additional credentials into their XMPP client specifically for the file upload.
This XEP defines an approach to request permissions from another entity to upload a file to a specific path on an HTTP server and at the same time receive an URL from which that file can later be downloaded again. These tuples consisting of a PUT and a GET-URL are called slots.

2 Requirements

- Be as easy to implement as possible. This is grounded on the idea that most programming languages already have HTTP libraries available.
- Be agnostic toward the distribution of the actual URL. Users can choose to send the URL in the body of a message stanza, utilize Out-of-Band Data (XEP-0066) [4], Jingle HTTP Transport Method (XEP-0370) [5], or even use it as their avatar in User Avatar (XEP-0084) [6].
- Anyone who knows the URL SHOULD be able to access it.

3 Discovering Support

An entity advertises support for this protocol by including the "urn:xmpp:http:upload:0" in its service discovery information features as specified in Service Discovery (XEP-0030) [7] or section 6.3 of Entity Capabilities (XEP-0115) [8]. To avoid unnecessary round trips an entity SHOULD also include the maximum file size as specified in Service Discovery Extensions

[1] XEP-0096: SI File Transfer <https://xmpp.org/extensions/xep-0096.html>.
[2] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.
[3] XEP-0313: Message Archive Management <https://xmpp.org/extensions/xep-0313.html>.
[4] XEP-0066: Out of Band Data <https://xmpp.org/extensions/xep-0066.html>.
[5] XEP-0370: Jingle HTTP Transport Method <https://xmpp.org/extensions/xep-0370.html>.
[6] XEP-0084: User Avatar <https://xmpp.org/extensions/xep-0084.html>.
[7] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[8] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

(XEP-0128) [9] if such a limitation exists. The field name MUST be ”max-file-size” and the value MUST be in bytes.
A user’s server SHOULD include any known entities that provide such services into its service discovery items.

Listing 1: Client sends service discovery request to server

```
<iq from='romeo@montague.tld/garden'
    id='step_01'
    to='montague.tld'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#items'/>
</iq>
```

Listing 2: Server replies to service discovery request

```
<iq from='montague.tld'
    id='step_01'
    to='romeo@montague.tld/garden'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#items'>
    <item jid='upload.montague.tld' name='HTTP␣File␣Upload' />
    <item jid='conference.montague.tld' name='Chatroom␣Service' />
  </query>
</iq>
```

Listing 3: Client sends service discovery request to upload service

```
<iq from='romeo@montague.tld/garden'
    id='step_02'
    to='upload.montague.tld'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 4: Upload service replies to service discovery request and reports a maximum file size of 5MiB

```
<iq from='upload.montague.tld'
    id='step_02'
    to='romeo@montague.tld/garden'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='store'
              type='file'
              name='HTTP␣File␣Upload' />
    <feature var='urn:xmpp:http:upload:0' />
```

[9] XEP-0128: Service Discovery Extensions <https://xmpp.org/extensions/xep-0128.html>.

```
    <x type='result' xmlns='jabber:x:data'>
      <field var='FORM_TYPE' type='hidden'>
        <value>urn:xmpp:http:upload:0</value>
      </field>
      <field var='max-file-size'>
        <value>5242880</value>
      </field>
    </x>
  </query>
</iq>
```

4 Requesting a slot

A client requests a new upload slot by sending an IQ-get to the upload service containing a <request> child element qualified by the urn:xmpp:http:upload:0 namespace. This element MUST include the attributes filename and size containing the file name and size (in bytes) respectively.
An additional attribute content-type containing the Content-Type is OPTIONAL.

Listing 5: Client requests a slot on the upload service

```
<iq from='romeo@montague.tld/garden'
    id='step_03'
    to='upload.montague.tld'
    type='get'>
  <request xmlns='urn:xmpp:http:upload:0'
    filename='très␣cool.jpg'
    size='23456'
    content-type='image/jpeg' />
</iq>
```

The upload service responds with both a PUT and a GET URL wrapped by a <slot> element. The service SHOULD keep the file name and especially the file ending intact. Using the same hostname for PUT and GET is OPTIONAL. The host MUST provide Transport Layer Security (RFC 5246 [10]). Both HTTPS URLs MUST adhere to RFC 3986 [11]. Non ASCII characters MUST be percent-encoded.
The <put> element MAY also contain a number of <header> elements which correspond to HTTP header fields. Each <header> element MUST have a name-attribute and a content with the value of the header. Only the following header names are allowed: Authorization, Cookie, Expires. The allowed headers provided in the response MUST be included in the HTTP PUT request. Other header names MUST be ignored by the requesting entity and MUST NOT be included in the HTTP request. The requesting entity MUST strip any newline characters from the header name and value before performing the HTTP request, but MUST preserve the

[10] RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 <http://tools.ietf.org/html/rfc5246>.
[11] RFC 3986: Uniform Resource Identifiers (URI): Generic Syntax <http://tools.ietf.org/html/rfc3986>.

relative order of multiple values for the same header in the request. Each header name MAY be present zero or more times, and are case insensitive (eXpires is the same as Expires).

Listing 6: The upload service responds with a slot

```
<iq from='upload.montague.tld'
    id='step_03'
    to='romeo@montague.tld/garden'
    type='result'>
  <slot xmlns='urn:xmpp:http:upload:0'>
    <put url='https://upload.montague.tld/4a771ac1-f0b2-4a4a-9700-
       f2a26fa2bb67/tr%C3%A8s%20cool.jpg'>
      <header name='Authorization'>Basic Base64String==</header>
      <header name='Cookie'>foo=bar; user=romeo</header>
    </put>
    <get url='https://download.montague.tld/4a771ac1-f0b2-4a4a-9700-
       f2a26fa2bb67/tr%C3%A8s%20cool.jpg' />
  </slot>
</iq>
```

5 Upload purpose

Retention policy is out of scope for this document. However, common retention strategies include enforcing user quotas that trigger the deletion of the oldest files when exceeded, and automatically deleting files older than a specified timeframe. This retention policy works well for message attachments, which are downloaded once per recipient device, but it may not be suitable for files requiring longer availability, like user avatars or microblog post images. To alleviate this problem, the requesting entity MAY specify a purpose when requesting an upload slot. This allows the upload service to sort files into different 'buckets' and apply different retention periods to those buckets. If no purpose is specified, the service MUST assume 'message' as the default purpose.

5.1 Message

An entity advertises support for the 'message' purpose by including "urn:xmpp:http:upload:purpose:0#message" in its service discovery features.
The requesting entity indicates that the upload slot is meant to be used for messaging purposes by including an element 'message' qualified by the 'urn:xmpp:http:upload:purpose:0' namespace.
Note: As the 'message' purpose is the default, explicitly announcing the feature and including this purpose in the slot request is technically redundant and is done solely for the sake of completeness.

Listing 7: Client requests a slot for the purpose of sending it as a message

```
<iq from='romeo@montague.tld/garden'
    id='step_03'
    to='upload.montague.tld'
    type='get'>
  <request xmlns='urn:xmpp:http:upload:0'
           filename='hi.jpg'
           size='23425'
           content-type='image/jpeg'>
    <message xmlns="urn:xmpp:http:upload:purpose:0" />
  </request>
</iq>
```

5.2 Profile

An entity advertises support for the 'profile' purpose by including "urn:xmpp:http:upload:purpose:0#profile" in its service discovery features.
The requesting entity indicates that the upload slot is meant to be used for profile purposes by including a 'profile' element, qualified by the urn:xmpp:http:upload:purpose:0 namespace. This purpose is for files, such as avatars (User Avatar (XEP-0084) [12]) or cover photos, that require longer retention, are smaller in size, and need a significantly lower overall quota.

Listing 8: Client requests an HTTP upload slot for the purpose of uploading an avatar

```
<iq from='romeo@montague.tld/garden'
    id='step_03'
    to='upload.montague.tld'
    type='get'>
  <request xmlns='urn:xmpp:http:upload:0'
           filename='avatar.jpg'
           size='100'
           content-type='image/jpeg'>
    <profile xmlns="urn:xmpp:http:upload:purpose:0"/>
  </request>
</iq>
```

5.3 Ephemeral

An entity advertises support for the ephemeral purpose by including "urn:xmpp:http:upload:purpose:0#ephemeral" in its service discovery features.
The requesting entity indicates that the upload slot is meant to be used for ephemeral purposes by including an 'ephemeral' element, qualified by the urn:xmpp:http:upload:purpose:0 namespace. The element MUST have an attribute called 'expire-before' that contains a DateTime as specified in XMPP Date and Time Profiles (XEP-0082) [13]. This purpose is similar

[12] XEP-0084: User Avatar <https://xmpp.org/extensions/xep-0084.html>.
[13] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.

to the default 'message' purpose but imposes an upper limit on the retention period, as specified by the requesting entity. The upload service MAY delete the file earlier based on its own retention policies, but MUST NOT make the file available after the time specified in the expire-before attribute.Example use cases for this purpose include, but are not limited to Ephemeral Messages (XEP-0466) [14] or 'Stories' that expire after 24 hours.

Listing 9: Client requests a slot with a limited lifetime

```
<iq from='romeo@montague.tld/garden'
    id='step_03'
    to='upload.montague.tld'
    type='get'>
  <request xmlns='urn:xmpp:http:upload:0'
           filename='eiffel-tower-status-picture.jpg'
           size='100'
           content-type='image/jpeg'>
    <ephemeral xmlns="urn:xmpp:http:upload:purpose:0"
               expire-before="2025-09-10T23:08:25Z" />
  </request>
</iq>
```

5.4 Permanent

An entity advertises support for the permanent purpose by including "urn:xmpp:http:upload:purpose:0#permanent" in its service discovery features.
The requesting entity indicates that the upload slot is meant for long term storage by including a 'permanent' element, qualified by the urn:xmpp:http:upload:purpose:0 namespace.Example use cases for this purpose include, but are not limited to Microblogging Over XMPP (XEP-0277) [15] and PubSub Social Feed (XEP-0472) [16].Support for this purpose - like any other specific purpose - is OPTIONAL.

Listing 10: Client requests a slot for permanent storage

```
<iq from='romeo@montague.tld/garden'
    id='step_03'
    to='upload.montague.tld'
    type='get'>
  <request xmlns='urn:xmpp:http:upload:0'
           filename='pubsub-blog-picture.jpg'
           size='42000'
           content-type='image/jpeg'>
    <permanent xmlns="urn:xmpp:http:upload:purpose:0" />
  </request>
```

[14] XEP-0466: Ephemeral Messages <https://xmpp.org/extensions/xep-0466.html>.
[15] XEP-0277: Microblogging over XMPP <https://xmpp.org/extensions/xep-0277.html>.
[16] XEP-0472: PubSub Social Feed <https://xmpp.org/extensions/xep-0472.html>.

```
</iq>
```

6 Error conditions

Instead of providing the client with a slot the service MAY respond with an error if the requested file size is too large. In addition the entity MAY inform the requester about the maximum file size.

Listing 11: Alternative response by the upload service if the file size was too large

```
<iq from='upload.montague.tld'
    id='step_03'
    to='romeo@montague.tld/garden'
    type='error'>
  <request xmlns='urn:xmpp:http:upload:0'
    filename='très␣cool.jpg'
    size='23456'
    content-type='image/jpeg' />
  <error type='modify'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' />
    <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>File too large.
       The maximum file size is 20000 bytes</text>
    <file-too-large xmlns='urn:xmpp:http:upload:0'>
      <max-file-size>20000</max-file-size>
    </file-too-large>
  </error>
</iq>
```

For any other type of error the service SHOULD respond with appropriate error types to indicate temporary or permanent errors.
For temporary errors such as exceeding a personal quota the service MAY include a <retry/> element qualified by the urn:xmpp:http:upload:0 namespace as a child of the <error/> element. The retry element MUST include an attribute 'stamp' which indicates the time at which the requesting entity may try again. The format of the timestamp MUST adhere to the date-time format specified in XMPP Date and Time Profiles (XEP-0082) [17] and MUST be expressed in UTC.

Listing 12: Alternative response by the upload service to indicate a temporary error after the client exceeded a quota

```
<iq from='upload.montague.tld'
    id='step_03'
    to='romeo@montague.tld/garden'
    type='error'>
  <request xmlns='urn:xmpp:http:upload:0'
    filename='très␣cool.jpg'
```

[17] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.

```
    size='23456'
    content-type='image/jpeg' />
  <error type='wait'>
    <resource-constraint xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' /
       >
    <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>Quota reached.
       You can only upload 5 files in 5 minutes</text>
    <retry xmlns='urn:xmpp:http:upload:0'
      stamp='2017-12-03T23:42:05Z' />
  </error>
</iq>
```

Listing 13: Alternative response by the upload service to indicate an auth error to a client that is not allowed to upload files

```
<iq from='upload.montague.tld'
    id='step_03'
    to='romeo@montague.tld/garden'
    type='error'>
  <request xmlns='urn:xmpp:http:upload:0'
     filename='très␣cool.jpg'
     size='23456'
     content-type='image/jpeg' />
  <error type='auth'>
    <forbidden xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' />
    <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>Only premium
       members are allowed to upload files</text>
  </error>
</iq>
```

7 Upload

The actual upload of the file happens via HTTP-PUT and is out of scope of this document. The upload service MUST reject the file upload if the Content-Length does not match the size of the slot request. The service SHOULD reject the file if the Content-Type has been specified beforehand and does not match. The service MAY assume application/octet-stream as a Content-Type if it the client did not specify a Content-Type at all.
In addition to the Content-Length and Content-Type header the client MUST include all allowed headers that came with the slot assignment.
There is no further XMPP communication required between the upload service and the client. A HTTP status Code of 201 means that the server is now ready to serve the file via the provided GET URL. If the upload fails for whatever reasons the client MAY request a new slot.

8 Implementation Notes

The upload service SHOULD choose an appropriate timeout for the validity of the PUT URL. Since there is no reason for a client to wait between requesting the slot and starting the upload, relatively low timeout values of around 300s are RECOMMENDED.
To make HTTP Upload work in web clients (including those hosted on a different domain) the upload service SHOULD set appropriate CORS-Headers. The exact headers and values are out of scope of this document but may include: *Access-Control-Allow-Origin*, *Access-Control-Allow-Methods* and *Access-Control-Allow-Headers*. For HTTP upload services that use custom *Authorization* or *Cookie* request header the CORS-Header *Access-Control-Allow-Credentials* might also be of importance.

```
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: OPTIONS, HEAD, GET, PUT
Access-Control-Allow-Headers: Authorization, Content-Type
Access-Control-Allow-Credentials: true
```

Clients SHOULD NOT interpret headers and treat them as opaque.

9 Security Considerations

9.1 Server side

Note: This section is not normative; it may be updated when general web security recommendations change in the future.
It is recommended to run the HTTP upload domain used for GET requests in appropriate isolation from other HTTP based services to avoid user-generated, malicious scripts to be executed in the context of said services. Isolation techniques can include, but are not limited to, setting the *Content-Security-Policy*.

```
Content-Security-Policy: default-src 'none'; frame-ancestors 'none';
```

The provided policy will prohibit a browser from executing all active content from the HTTP upload domain (*default-src 'none'*) and forbid embedding it from other pages (*frame-ancestors 'none'*). More information on Content-Security-Policy can be found on infosec.mozilla.org.
Further isolation can be achieved by hosting those files on an entirely different domain instead of using subdomains.
Headers may be signed so that receiving HTTP entities can verify these haven't been tempered with by clients.

9.2 Uploader

- Requesting entities MUST strip any newline characters from the HTTP header names and values before making the PUT request.
- Requesting entities MUST ensure that only the headers that are explicitly allowed by this XEP (Authorization, Cookie, Expires) are copied from the slot response to the HTTP request.

9.3 General

- Service implementors SHOULD use long randomized parts in their URLs making it impossible to guess the location of arbitrary files.
- Implementors should keep in mind, that without additional end-to-end-encryption, files uploaded to a service described in this document may be stored in plain text. Client implementors are advised to either use this only for semi public files (for example files shared in a public MUC or a PEP Avatar) or implement appropriate end-to-end encryption.
- Up- and downloading files will leak the client's IP address to the HTTP service. The HTTP service might not be the same service as the XMPP service the client is currently connected to.

10 IANA Considerations

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

11 XMPP Registrar Considerations

11.1 Protocol Namespaces

This specification defines the following XML namespace:

- urn:xmpp:http:upload:0

Upon advancement of this specification from a status of Experimental to a status of Draft, the XMPP Registrar [18] 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) [19].

[18] 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/>.

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

12 XML Schema

```
<xml version="1.0" encoding="utf8">
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:xmpp:http:upload:0"
    xmlns="urn:xmpp:http:upload:0">
  <xs:element name="request">
    <xs:complexType>
      <xs:attribute name="filename" type="xs:string" use="required"/>
      <xs:attribute name="size" type="xs:positiveInteger" use="
          required"/>
      <xs:attribute name="content-type" type="xs:string" use="optional
          "/>
    </xs:complexType>
  </xs:element>
  <xs:element name="slot">
    <xs:complexType>
     <xs:sequence>
       <xs:element name="put" minOccurs="1" maxOccurs="1">
         <xs:complexType>
           <xs:attribute name="url" type="xs:string" use="required"/>
           <xs:sequence>
             <xs:element name="header" minOccurs="0" maxOccurs="
                 unbounded" type="xs:string">
               <xs:complexType>
                 <xs:attribute name="name" use="required">
                   <xs:simpleType>
                     <xs:restriction base="xs:string">
                       <xs:enumeration value="Authorization"/>
                       <xs:enumeration value="Cookie"/>
                       <xs:enumeration value="Expires"/>
                     </xs:restriction>
                   </xs:simpleType>
                 </xs:attribute>
               </xs:complexType>
             </xs:element>
           </xs:sequence>
         </xs:complexType>
       </xs:element>
       <xs:element name="get" minOccurs="1" maxOccurs="1">
         <xs:complexType>
           <xs:attribute name="url" type="xs:string" use="required"/>
         </xs:complexType>
       </xs:element>
     </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="file-too-large">
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element name="max-file-size" type="xs:positiveInteger"
            minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="retry">
    <xs:complexType>
      <xs:attribute name="stamp" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```