XEP-0065: SOCKS5 Bytestreams

Dave Smith
mailto:dizzyd@jabber.org
xmpp:dizzyd@jabber.org

Matthew Miller
mailto:linuxwolf@outer-planes.net
xmpp:linuxwolf@outer-planes.net

Peter Saint-Andre
mailto:xsf@stpeter.im
xmpp:peter@jabber.org
http://stpeter.im/

Justin Karneges
mailto:justin@affinix.com
xmpp:justin@andbit.net

2015-09-17
Version 1.8.1

Status	Type	Short Name
Draft	Standards Track	bytestreams

This document defines an XMPP protocol extension for establishing an out-of-band bytestream between any two XMPP users, mainly for the purpose of file transfer. The bytestream can be either direct (peer-to-peer) or mediated (though a special-purpose proxy server). The typical transport protocol used is TCP, although UDP can optionally be supported as well.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

XMPP is designed for sending relatively small chunks of XML between network entities and is not designed for sending binary data. However, sometimes it is desirable to send binary data to another entity that one has discovered on the XMPP network (e.g., to send a file). Therefore it is valuable to have a generic protocol for streaming binary data between any two entities on an XMPP network. The main application for such a bytestreaming technology is file transfer as specified in SI File Transfer (XEP-0096) [1] and Jingle File Transfer (XEP-0234) [2]. However, other applications are possible, which is why it is important to develop a generic protocol rather than one that is specialized for a particular application such as file transfer. This document defines a protocol that meets the following conditions:

- Bytestreams are established over standard TCP connections (RFC 793 [3]) or UDP associations (RFC 768 [4]), where TCP support is REQUIRED and UDP support is OPTIONAL
- Sockets can be direct (peer-to-peer) or mediated (established through a relay)
- Where possible, standard wire protocols are used

Specifically, this protocol makes use of the SOCKS 5 protocol, which is an IETF-approved, IPv6-ready technology for bytestreams defined in RFC 1928 [5]. However, because this protocol uses a subset of the SOCKS5 protocol that is specially adapted for bytestreaming over XMPP, existing SOCKS5 proxies cannot be used to implement this protocol without modifications. There are two scenarios addressed by this protocol:

1. A direct connection in which the StreamHost is the Requester, as described under Direct Connection
2. A mediated connection in which the StreamHost is a Proxy, as described under Mediated Connection

Early versions of this specification documented only the use of TCP connections. In version 1.6 (approved in November 2004), optional UDP associations were added, as described in the Optional UDP Support section of this document. However, the main body of this document describes the use of TCP, which is the primary method of SOCKS5 Bytestreams ("S5B").

2 Terminology

The following terms are used throughout this document.

[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] RFC 793: Transmission Control Protocol <http://tools.ietf.org/html/rfc0793>.
[4] RFC 768: User Datagram Protocol <http://tools.ietf.org/html/rfc0768>.
[5] RFC 1928: SOCKS Protocol Version 5 <http://tools.ietf.org/html/rfc1928>.

Requester The entity that starts a bytestream negotiation with a Target. Before version 1.8 of this document a Requester was known as an Initiator.

Target The entity with which the Requester is attempting to establish a bytestream.

Proxy An entity that is willing to be a middleman for the bytestream between the Requester and the Target.

StreamHost The system that the Target connects to and that is "hosting" the bytestream; the Streamhost can be either the Requester or a Proxy.

StreamID A relatively unique Stream ID for this connection; this is generated by the Requester for tracking purposes.

Note: Because either party can attempt to establish a bytestream (this is formalized in Jingle SOCKS5 Bytestreams Transport Method (XEP-0260) [6]), the Requester and the Target roles apply to a particular S5B negotiation, and do not map to the Initiator and Responder roles from Jingle (XEP-0166) [7] in a fixed way. For example, during a Jingle negotiation the Jingle Initiator might first take on the role of an S5B Requester (with the Jingle Responder being the S5B Target) but if that first bytestreams negotiation fails (the so-called "fallback scenario") then the Jingle Responder might take on the role of an S5B Requester (with the Jingle Initiator being the S5B Target).
In the protocol flow diagrams, the line types have the following meaning:

- "----" ... communications over XMPP
- "____" ... communications over TCP
- "\\\\" and "////" ... communications over SOCKS5
- "====" ... communications over the bytestream itself

In the examples, "streamer.example.com" is a Proxy that services bytestreams on port 7625.

3 Determining Support

If an entity supports this protocol, it MUST advertise that fact in its responses to Service Discovery (XEP-0030) [8] information ("disco#info") requests by returning a feature of "http://jabber.org/protocol/bytestreams".

[6] XEP-0260: Jingle SOCKS5 Bytestreams Transport Method <https://xmpp.org/extensions/xep-0260.html>.
[7] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.
[8] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

Listing 1: Requester Sends Service Discovery Request to Target

```
<iq from='requester@example.com/foo'
    id='gr91cs53'
    to='target@example.org/bar'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 2: Target Replies to Service Discovery Request

```
<iq from='target@example.org/bar'
    id='gr91cs53'
    to='requester@example.com/foo'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='client' type='pc'/>
    <feature var='http://jabber.org/protocol/bytestreams'/>
  </query>
</iq>
```

4 Discovering Proxies

Before attempting to initiate a bytestream, the Requester might need to find a proxy (e.g., if it has not been configured to know about a proxy). It can do so using Service Discovery by communicating with its server.

Listing 3: Requester Sends Service Discovery Request to Server

```
<iq from='requester@example.com/foo'
    id='pi2b15fv'
    to='example.com'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#items'/>
</iq>
```

The server will return all of the items it knows about.

Listing 4: Server Replies to Service Discovery Request

```
<iq from='example.com'
    id='pi2b15fv'
    to='requester@example.com/foo'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#items'>
    <item jid='chatrooms.example.com' name='Chatroom␣Service'/>
    <item jid='news.example.com' name='News␣Feeds'/>
    <item jid='streamer.example.com' name='File␣Transfer␣Relay'/>
```

```
  </query>
</iq>
```

In this case, the ”streamer.example.com” is a bytestreams proxy.
For each item in the disco#items result, the Requester needs to query to determine if it is a bytestreams proxy.

Listing 5: Requester Sends Service Discovery Request to Proxy

```
<iq from='requester@example.com/foo'
    id='yx92b153'
    to='streamer.example.com'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

The proxy returns its information and the Requester inspects it to determine if it contains an identity of category ”proxy” and type ”bytestreams”.

Listing 6: Server Replies to Service Discovery Request

```
<iq from='streamer.example.com'
    id='yx92b153'
    to='requester@example.com/foo'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='proxy'
              type='bytestreams'
              name='File␣Transfer␣Relay'/>
    <feature var='http://jabber.org/protocol/bytestreams'/>
  </query>
</iq>
```

Next the Requester needs to request the full network address to be used for bytestreaming through the Proxy. This is done by sending an IQ-get to the proxy containing a <query/> element qualified by the bytestreams namespace (not the service discovery namespace). [9]

Listing 7: Requester Requests Network Address from Proxy

```
<iq from='requester@example.com/foo'
    id='uj2c15z9'
    to='streamer.example.com'
    type='get'>
```

[9] Before version 1.8 of this specification, the <query/> element in this use case possessed a 'sid' attribute; however, it is unnecessary for the Requester to specify the StreamID here and it would be harmful for the Proxy to reserve the StreamID at this point because the StreamID might never be used (thus forcing the Proxy to establish and maintain state about the bytestream) and because the Requester might use the Proxy's services for multiple different streams.

```
  <query xmlns='http://jabber.org/protocol/bytestreams'/>
</iq>
```

The Proxy replies by returning an IQ-result that contains its network address, structured using the <streamhost/> child of the <query/> element; the <streamhost/> element MUST possess the following attributes:

- host = the IP address or DNS domain name of the StreamHost for SOCKS5 communication over TCP (if the value is an IPv6 address, it MUST be formatted according to RFC 5952 [10], as is done in XMPP Core [11])
- jid = the JabberID of the StreamHost for communication over XMPP
- port = the port on which to connect for SOCKS5 communication over TCP

Note: If the value of the 'host' attribute is a DNS domain name, it MUST be resolvable to the IP address on which the Proxy (or an instance thereof) is hosted using an A or AAAA lookup.

Listing 8: Proxy Informs Requester of Network Address

```
<iq from='streamer.example.com'
    id='uj2c15z9'
    to='requester@example.com/foo'
    type='result'>
  <query xmlns='http://jabber.org/protocol/bytestreams'>
    <streamhost
        host='24.24.24.1'
        jid='streamer.example.com'
        port='7625'/>
  </query>
</iq>
```

If the Requester does not have permissions to initiate bytestreams on the Proxy for whatever reason (e.g., a proxy implementation might enable administrators to ban JIDs or domains from using the Proxy), the Proxy MUST return a <forbidden/> error to the Requester.

Listing 9: Requester is Forbidden to use Proxy

```
<iq from='streamer.example.com'
    id='uj2c15z9'
    to='requester@example.com/foo'
    type='error'>
  <error type='auth'>
    <forbidden
        xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
```

[10] RFC 5952: A Recommendation for IPv6 Address Text Representation <http://tools.ietf.org/html/rfc5952>.
[11] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

```
  </error>
</iq>
```

If the Proxy is unable to act as a StreamHost, the Proxy MUST return an error to the Requester, which SHOULD be <not-allowed/>.

Listing 10: Proxy is Unable to Act as a StreamHost

```
<iq from='requester@example.com/foo'
    id='uj2c15z9'
    to='streamer.example.com'
    type='error'>
  <error type='cancel'>
    <not-allowed
        xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

5 Direct Connection

In this situation, the StreamHost is the Requester, which means that the Requester knows the network address of the StreamHost and knows when to activate the bytestream.

5.1 Process

For direct connections, the process for establishing a bytestream is as follows:

1. Requester initiates S5B negotiation with Target by sending an IQ-set that includes the full JID <localpart@domain.tld/resource> and network address of StreamHost/Requester as well as the StreamID (SID) of the proposed bytestream (and, optionally, the calculated DST.ADDR value; see under Use with Multi-User Chat).

2. Target opens a TCP socket to the specified network address at the StreamHost/Requester.

3. Target requests SOCKS5 connection at StreamHost/Requester.

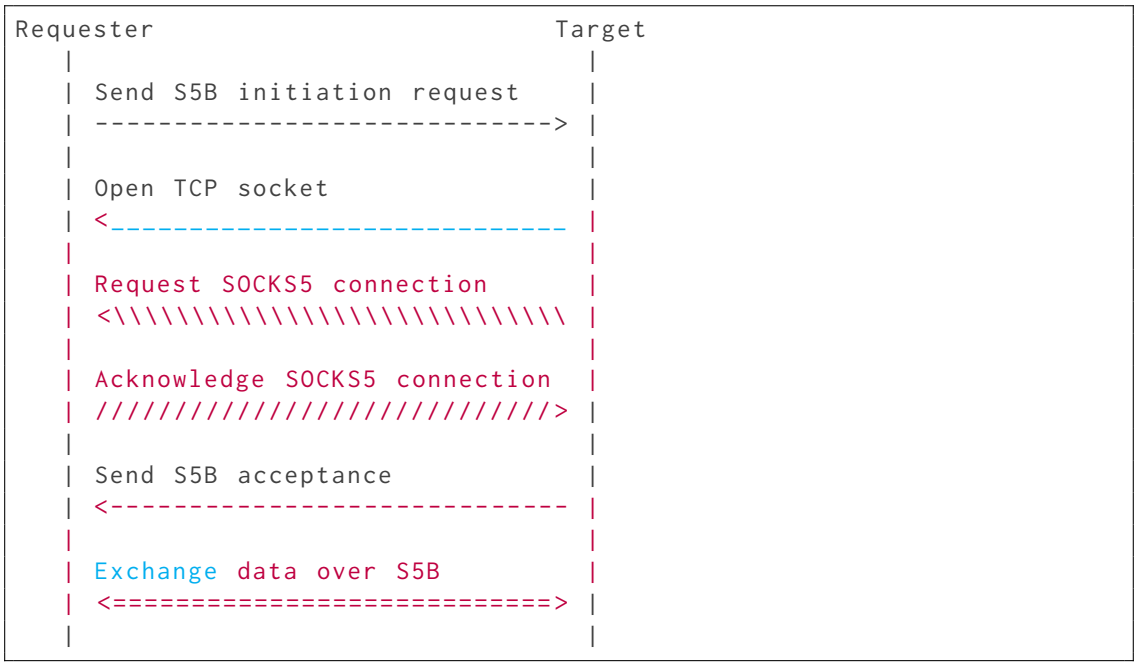
4. StreamHost/Requester sends acknowledgement of successful connection to Target via SOCKS5.

5. Target accepts the S5B stream by returning an IQ-result to the Requester, preserving the 'id' of the initial IQ-set.

6. Requester and Target exchange data over the bytestream.

5.2 Flow

The data flow is shown in the following diagram.



5.3 Protocol

5.3.1 Requester Initiates S5B Negotiation

To initiate an S5B negotiation with the Target, the Requester sends network address information about one or more StreamHosts to the Target. In the case of a direct connection, the Requester might include information only about itself (as shown in the following example) or about itself and a Proxy.
The <query/> element MUST contain one or more <streamhost/> elements, each of which MUST possess the 'host', 'jid', and 'port' attributes. The <query/> element MUST possess a 'sid' attribute that specifies the Stream ID for this bytestream. The <query/> element MAY possess a 'mode' attribute whose value is "tcp" (the default) or "udp" (for which see Optional UDP Support). The <query/> element MAY possess a 'dstaddr' attribute whose value is the Requester's calculated hash value for the SOCKS5 DST.ADDR field (see Use with Multi-User

Chat).

Listing 11: Requester Initiates Negotiation

```
<iq from='requester@example.com/foo'
    id='hu3vax16'
    to='target@example.org/bar'
    type='set'>
  <query xmlns='http://jabber.org/protocol/bytestreams'
         sid='vxf9n471bn46'>
    <streamhost
        jid='requester@example.com/foo'
        host='192.168.4.1'
        port='5086'/>
  </query>
</iq>
```

If the request is malformed (e.g., the <query/> element does not include the 'sid' attribute), the Target MUST return an error of <bad-request/>.
Else if the Target is unwilling to accept the bytestream, it MUST return an error of <not-acceptable/> to the Requester.

Listing 12: Target Refuses Bytestream

```
<iq from='target@example.org/bar'
    id='hu3vax16'
    to='requester@example.com/foo'
    type='error'>
  <error type='modify'>
    <not-acceptable
        xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the Target is willing to negotiate a bytestream, it proceeds as shown in the following sections.

5.3.2 Target Establishes SOCKS5 Connection with StreamHost/Requester

Next the Target attempts to open a standard TCP socket on the network address of the StreamHost/Requester (for information about UDP usage, see the Optional UDP Support section of this document).
Note: If the Requester provides more than one StreamHost, the Target SHOULD try to connect to them in the order of the <streamhost/> children within the <query/> element. Jingle SOCKS5 Bytestreams Transport Method (XEP-0260) [12] modifies this rule by providing explicit priorities for each streamhost candidate.

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

If the Target is able to open a TCP socket on a StreamHost/Requester, it MUST use the SOCKS5 protocol to establish a SOCKS5 connection. In accordance with RFC 1928, the Target might need to authenticate in order to use the proxy. However, any authentication required is beyond the scope of this document.
Once the Target has successfully authenticated with the StreamHost/Requester, it sends a CONNECT request (CMD = X'01') in order to continue the negotiation. The following rules apply:

1. The hostname MUST be SHA1(SID + Requester JID + Target JID) where the definition of the SHA1 hashing algorithm is as specified by RFC 3174 [13] and the output is hexadecimal-encoded (not binary); as noted above and under Use with Multi-User Chat, the DST.ADDR value might have been provided directly from the Requester to the Target).
2. The port MUST be 0 (zero).
3. The JIDs used as input to the hash function MUST be the actual JIDs used for the IQ exchange between the Requester and the Target (these might be full JIDs (<localpart@domain.tld/resource> or <domain.tld/resource>) or bare JIDs (<localpart@domain.tld> or <domain.tld>) depending on the addresses of the entities involved in the negotiation).
4. The appropriate stringprep profiles (as specified in RFC 6122 [14]) MUST be applied to the JIDs before application of the SHA1 hashing algorithm.

Listing 13: Target Establishes SOCKS5 Connection with StreamHost

```
CMD = X'01'
ATYP = X'03'
DST.ADDR = SHA1 Hash of: (SID + Requester JID + Target JID)
DST.PORT = 0
```

Listing 14: StreamHost Acknowledges Connection

```
STATUS = X'00'
```

When replying to the Target in accordance with Section 6 of RFC 1928, the StreamHost MUST set the BND.ADDR and BND.PORT to the DST.ADDR and DST.PORT values provided by the client in the connection request.
If the Target tries but is unable to connect to any of the StreamHosts and it does not wish to attempt a connection from its side, it MUST return an <item-not-found/> error to the Requester.

[13] RFC 3174: US Secure Hash Algorithm 1 (SHA1) <http://tools.ietf.org/html/rfc3174>.
[14] RFC 6122: Extensible Messaging and Presence Protocol (XMPP): Address Format <http://tools.ietf.org/html/rfc6122>.

Listing 15: Target Is Unable to Connect to Any StreamHost and Wishes to End Negotiation

```
<iq from='target@example.org/bar'
    id='hu3vax16'
    to='requester@example.com/foo'
    type='error'>
  <error type='cancel'>
    <item-not-found
        xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

5.3.3 Target Acknowledges Bytestream

After the Target has authenticated with the StreamHost/Requester, it replies to the initiate request with an IQ-result whose <query/> element contains a <streamhost-used/> child that specifies which StreamHost was used (in this case, the StreamHost/Requester).

Listing 16: Target Notifies Requester of Bytestream

```
<iq from='target@example.org/bar'
    id='hu3vax16'
    to='requester@example.com/foo'
    type='result'>
  <query xmlns='http://jabber.org/protocol/bytestreams'
         sid='vxf9n471bn46'>
    <streamhost-used jid='requester@example.com/foo'/>
  </query>
</iq>
```

At this point, the Requester knows which StreamHost was used by the Target and the parties are able to use the StreamHost/Requester to exchange data over the bytestream.

6 Mediated Connection

In this situation, the StreamHost is not the Requester but a Proxy, which means that the Requester needs to discover the network address of the StreamHost before sending the initiation request to the Target, needs to negotiate a connection with the StreamHost in the same way that the Target does, and needs to ask the StreamHost to activate the bytestream before it can be used.

6.1 Process

For mediated connections, the process for establishing a bytestream is as follows:

1. As a precondition, the Requester optionally discovers the network address of StreamHost over XMPP as discussed in the Service Discovery section of this document.

2. Requester initiates S5B negotation with Target by sending IQ-set that includes the JabberID and network address of StreamHost as well as the StreamID (SID) of the proposed bytestream (and, optionally, the calculated DST.ADDR value; see under Use with Multi-User Chat).

3. Target opens a TCP socket to the selected StreamHost.

4. Target requests SOCKS5 connection at StreamHost/Proxy.

5. StreamHost sends acknowledgement of successful connection to Target via SOCKS5.

6. Target sends IQ-result to Requester, preserving the 'id' of the initial IQ-set.

7. Requester opens a TCP socket at the StreamHost.

8. Requester establishes connection via SOCKS5, with the DST.ADDR and DST.PORT parameters set to the values defined below.

9. StreamHost sends acknowledgement of successful connection to Requester via SOCKS5.

10. Requester sends IQ-set to StreamHost requesting that StreamHost activate the bytestream associated with the StreamID.

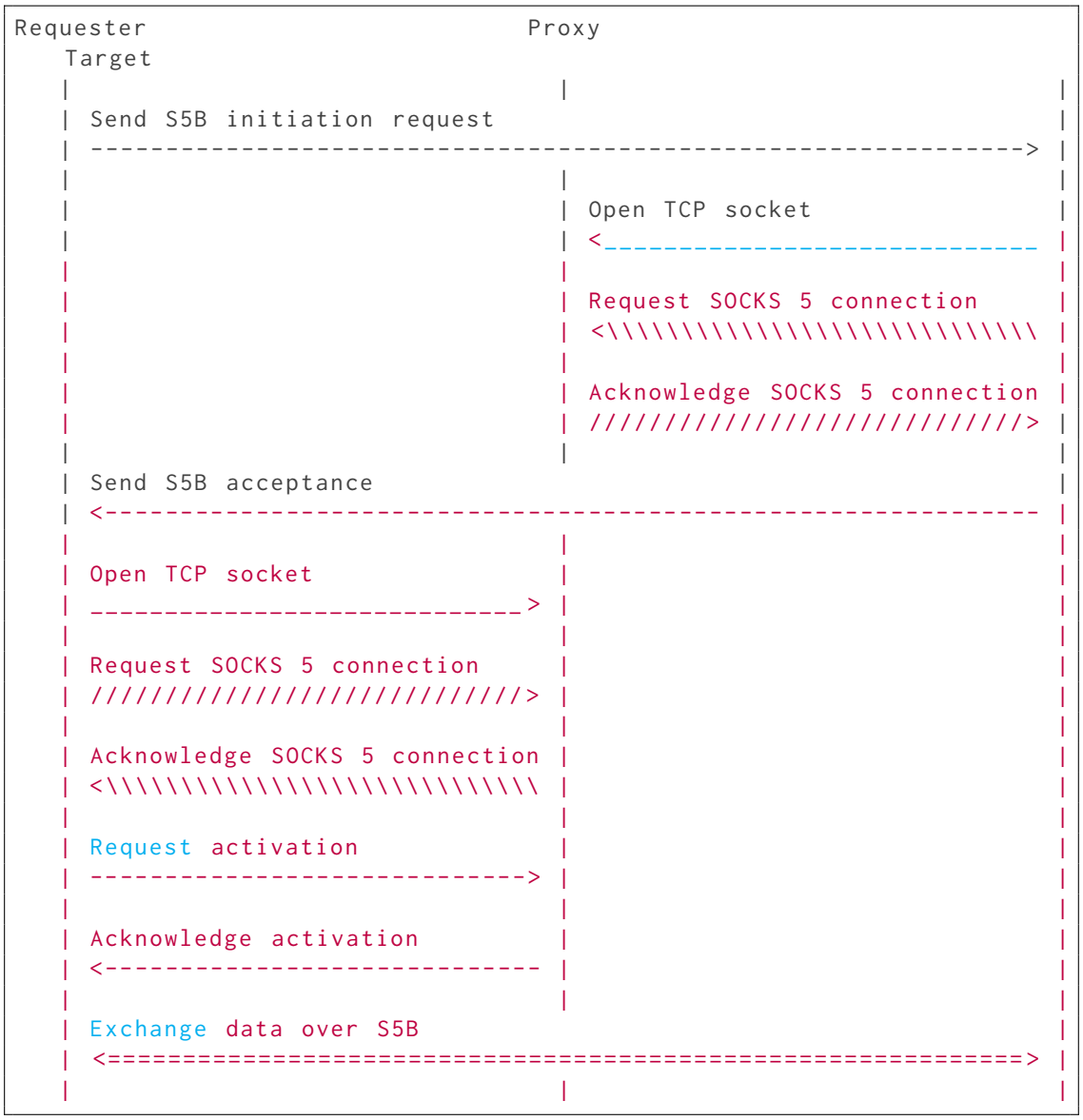
11. StreamHost activates the bytestream. (Data is now relayed between the two SOCKS5 connections by the proxy.)

12. StreamHost sends IQ-result to Requester acknowledging that the bytestream has been activated (or specifying an error).

13. Requester and Target can begin using the bytestream.

6.2 Flow

The data flow is shown in the following diagram.



6.3 Protocol

6.3.1 Requester Initiates S5B Negotiation

To initiate an S5B negotiation with the Target, the Requester sends network address information about one or more StreamHosts to the Target. In the case of a mediated connection, the Requester might include information only about the Proxy (as shown in the following

example) or about the Proxy and itself.
The <query/> element MUST contain one or more <streamhost/> elements, each of which MUST possess the 'host', 'jid', and 'port' attributes. The <query/> element MUST possess a 'sid' attribute that specifies the Stream ID for this bytestream. The <query/> element MAY possess a 'mode' attribute whose value is "tcp" (the default) or "udp" (for which see Optional UDP Support). The <query/> element MAY possess a 'dstaddr' attribute whose value is the Requester's calculated hash value for the SOCKS5 DST.ADDR field (see Use with Multi-User Chat).

Listing 17: Requester Initiates Negotiation

```
<iq from='requester@example.com/foo'
    id='npq71g53'
    to='target@example.org/bar'
    type='set'>
  <query xmlns='http://jabber.org/protocol/bytestreams'
         sid='vxf9n471bn46'>
    <streamhost
        host='24.24.24.1'
        jid='streamer.example.com'
        port='7625'/>
  </query>
</iq>
```

If the Target is willing to negotiate a bytestream, it proceeds as shown in the following sections.

6.3.2 Target Establishes SOCKS5 Connection with Proxy

Next the Target attempts to open a standard TCP socket on the network address of the Proxy. If the Target is able to open a TCP socket on the Proxy, it uses the SOCKS5 protocol to establish a SOCKS5 connection. In accordance with RFC 1928, the Target might need to authenticate in order to use the proxy. However, any authentication required is beyond the scope of this document.
Once the Target has successfully authenticated with the Proxy, it sends a CONNECT request (CMD = X'01') in order to continue the negotiation. The following rules apply:

1. The hostname MUST be SHA1(SID + Requester JID + Target JID) where the definition of the SHA1 hashing algorithm is as specified by RFC 3174 [15] and the output is hexadecimal-encoded (not binary); as noted above and under Use with Multi-User Chat, the DST.ADDR value might have been provided directly from the Requester to the Target).

2. The port MUST be 0 (zero).

[15] RFC 3174: US Secure Hash Algorithm 1 (SHA1) <http://tools.ietf.org/html/rfc3174>.

3. The JIDs provided MUST be the JIDs used for the IQ exchange between the Requester and the Target, which MAY be full JIDs (<localpart@domain.tld/resource> or <domain.tld/resource>) or bare JIDs (<localpart@domain.tld> or <domain.tld>).

4. The appropriate stringprep profiles (as specified in RFC 6122) MUST be applied to the JIDs before application of the SHA1 hashing algorithm.

Listing 18: Target Establishes SOCKS5 Connection with StreamHost

```
CMD = X'01'
ATYP = X'03'
DST.ADDR = SHA1 Hash of: (SID + Requester JID + Target JID)
DST.PORT = 0
```

Listing 19: StreamHost Acknowledges Connection

```
STATUS = X'00'
```

When replying to the Target in accordance with Section 6 of RFC 1928, the Proxy MUST set the BND.ADDR and BND.PORT to the DST.ADDR and DST.PORT values provided by the client in the connection request.

6.3.3 Target Acknowledges Bytestream

After the Target has established a SOCKS5 connection with the Proxy, it replies to the initiate request with an IQ-result whose <query/> element contains a <streamhost-used/> child that specifies which StreamHost was used (in this case, the Proxy).

Listing 20: Target Notifies Requester of Bytestream

```
<iq from='target@example.org/bar'
    id='npq71g53'
    to='requester@example.com/foo'
    type='result'>
  <query xmlns='http://jabber.org/protocol/bytestreams'
         sid='vxf9n471bn46'>
    <streamhost-used jid='streamer.example.com'/>
  </query>
</iq>
```

At this point, the Requester knows which StreamHost was used by the Target.

6.3.4 Requester Establishes SOCKS5 Connection with StreamHost

Here, unlike the direct connection case described above, the Requester also needs to establish a SOCKS5 connection to the Proxy before the parties are able to use the Proxy to exchange

data over the bytestream. Therefore the Requester will establish a connection to the SOCKS5 proxy in the same way the Target did (passing the same value for the CONNECT request), as shown in the following examples.

Listing 21: Requester Connects to StreamHost

```
CMD = X'01'
ATYP = X'03'
DST.ADDR = SHA1 Hash of: (SID + Requester JID + Target JID)
DST.PORT = 0
```

Listing 22: StreamHost Acknowledges Connection to Requester

```
STATUS = X'00'
```

6.3.5 Activation of Bytestream

Next the Requester needs to activate the bytestream with the Proxy. This is done by sending an IQ-set to the Proxy, including an <activate/> element whose XML character data specifies the full or bare JID of the Target.

Listing 23: Requester Requests Activation of Bytestream

```
<iq from='requester@example.com/foo'
    id='oqx6t1c9'
    to='streamer.example.com'
    type='set'>
  <query xmlns='http://jabber.org/protocol/bytestreams'
         sid='vxf9n471bn46'>
    <activate>target@example.org/bar</activate>
  </query>
</iq>
```

Using this information, with the SID and from address on the packet, the Proxy is able to activate the stream by hashing the SID + Requester JID + Target JID and comparing the result against the DST.ADDR it has received from the Target and Receiver. Although this provides a reasonable level of trust that the activation request came from the Requester, it does not guard against active or even passive attacks against the bytestreams negotiation (see the Security Considerations for information about potential hijacking of the negotiation).
If the Proxy can fulfill the request, it MUST respond to the Requester with an IQ-result.

Listing 24: Proxy Informs Requester of Activation

```
<iq from='streamer.example.com'
    id='oqx6t1c9'
    to='requester@example.com/foo'
    type='result'/>
```

At this point the parties can begin exchanging data over the bytestream.
If the Proxy cannot fulfill the request, it MUST return an IQ-error to the Requester; the following conditions are defined:

- <item-not-found/> if the 'from' address does not match that of the Requester's full JID
- <not-allowed/> if only one party (either Requester or Recipient, but not both) is connected to the Proxy
- <not-authorized/> if the hashes do not match
- <internal-server-error/> if the proxy cannot activate the bytestream because of some internal malfunction

7 Use with Multi-User Chat

When one occupant of a Multi-User Chat (XEP-0045) [16] conference sends an S5B invitation to another occupant, often the MUC room obscures the real JID of the Target from the Requester and the real JID of the Requester from the Target. This means that the two parties might not have the same view of the information needed to calculate the DST.ADDR. To overcome this problem, the Requester SHOULD calculate the DST.ADDR based on the SID, its real JID, and the room JID (room@host/nick) of the Target, then include the calculated hash as the value of a 'dstaddr' attribute on the <query/> element. The Requester then sends the IQ-set to the Target's room JID because it does not know the Target's real JID.
An example follows.

Listing 25: Requester Initiates Negotiation Through MUC Room

```
<iq from='requester@example.com/foo'
    id='npq71g53'
    to='room@conference.example.net/Tget'
    type='set'>
  <query xmlns='http://jabber.org/protocol/bytestreams'
         dstaddr='416781edf1ae50bad01cb8509ba35b43952bc345'
         sid='yia72g3v49j7'>
    <streamhost
        host='24.24.24.1'
        jid='streamer.example.com'
        port='7625'/>
  </query>
</iq>
```

The MUC room will then forward the IQ-set to the Target's real JID with a 'from' address of the Requester's room JID.

[16] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.

Listing 26: MUC Room Forwards Initiation Request

```
<iq from='room@conference.example.net/Rter'
    id='npq71g53'
    to='target@example.org/bar'
    type='set'>
  <query xmlns='http://jabber.org/protocol/bytestreams'
         dstaddr='416781edf1ae50bad01cb8509ba35b43952bc345'
         sid='yia72g3v49j7'>
    <streamhost
        host='24.24.24.1'
        jid='streamer.example.com'
        port='7625'/>
  </query>
</iq>
```

Now the parties can proceed as defined for the direct or mediated connection. See the Security Considerations for information about potential hijacking of the negotiation.

8 Optional UDP Support

Support for UDP associations is strictly OPTIONAL. However, implementations that support UDP associations MUST adhere to the profile described in this section.

8.1 Discovering UDP Support

If an implementation supports UDP associations, it MUST advertise that separately by returning a feature of 'http://jabber.org/protocol/bytestreams#udp' in response to Service Discovery information requests.

Listing 27: Requester Sends Service Discovery Request to Target

```
<iq from='requester@example.com/foo'
    id='pys51v35'
    to='target@example.org/bar'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

If the Target supports UDP associations, it MUST include a feature of 'http://jabber.org/protocol/bytestreams#udp' in the service discovery result.

Listing 28: Target Replies to Service Discovery Request

```
<iq from='target@example.org/bar'
    id='pys51v35'
```

```
    to='requester@example.com/foo'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity
        category='proxy'
        type='bytestreams'
        name='File Transfer Relay'/>
    <feature var='http://jabber.org/protocol/bytestreams'/>
    <feature var='http://jabber.org/protocol/bytestreams#udp'/>
  </query>
</iq>
```

8.2 Requesting UDP Mode

UDP associations are requested by setting the 'mode' attribute to a value of "udp" rather than "tcp".

Listing 29: Initiation of Interaction (UDP)

```
<iq from='requester@example.com/foo'
    id='xi2d1973'
    to='target@example.org/bar'
    type='set'>
  <query xmlns='http://jabber.org/protocol/bytestreams'
         mode='udp'
         sid='mySID'>
    <streamhost
        host='192.168.4.1'
        jid='requester@example.com/foo'
        port='5086'/>
  </query>
</iq>
```

8.3 UDP Process

There is one main difference between UDP mode and TCP mode: rather than simply establishing a TCP connection, the Target and/or Requester MUST (1) establish a UDP association and then (2) initialize the UDP channel. In particular:

- If direct connection is followed, Target MUST complete UDP association and initialization of the UDP channel before informing Requester of success via the <streamhost-used/> element.
- If mediated connection is followed, (1) Target MUST complete UDP association and initialization of the UDP channel before informing Requester of success via the

<streamhost-used/> element, and (2) Requester MUST complete UDP association and initialization of the UDP channel before asking StreamHost to activate the bytestream.

The processes for establishing the UDP association and for initializing the UDP channel are described below.

8.3.1 Establishing the UDP Association

Once the Target has successfully authenticated with the Proxy over TCP (as described under Target Establishes SOCKS5 Connection with StreamHost), it MUST send a UDP ASSOCIATE request (CMD = X'03') to the host identified by the algorithm defined above.

Listing 30: Target Requests UDP Association with StreamHost

```
CMD = X'03'
ATYP = X'03'
DST.ADDR = SHA1 Hash of: (SID + Requester JID + Target JID)
DST.PORT = 0
```

The StreamHost then acknowledges this request:

Listing 31: StreamHost Acknowledges Request

```
STATUS = X'00'
```

8.3.2 Initializing the UDP Channel

After connecting to the StreamHost, the Target (direct connection) or both Target and Requester (mediated connection) MUST initialize the UDP channel. In order to do so, each sending entity MUST send a SOCKS5 UDP packet to the StreamHost on the same port used for the initial TCP connection (in the foregeoing example, a host of 192.168.4.1 and port of 5086), with DST.PORT set to '1' and DATA containing the sending entity's JID (i.e, the JID of either the Target or Requester).

Listing 32: Target or Requester Sends UDP Initialization Packet to StreamHost

```
ATYP = X'03'
DST.ADDR = SHA1 Hash of: (SID + Requester JID + Target JID)
DST.PORT = 1
DATA = Target or Requester JID
```

Upon successful receipt by the StreamHost, the StreamHost MUST reply with a message notification indicating success:

Listing 33: StreamHost Notifies Target or Requester of UDP Success

```
<message
    from='streamer.example.com'
    to='target@example.org/bar'
    id='zy3v29h6'>
  <udpsuccess xmlns='http://jabber.org/protocol/bytestreams'
              dstaddr='Value of Hash'/>
</message>
```

The <udpsuccess/> element indicates that the StreamHost has received a UDP initialization packet. This element has a single attribute containing the DST.ADDR that was used in the UDP packet.
If Target is unable to initialize the UDP channel, it MUST return a <remote-server-not-found/> error to RequesteRequester.
Note: Since UDP is not reliable, the Target SHOULD resend the UDP packet if the reply notification is not received within a short time (a 5-second retry is RECOMMENDED). The StreamHost SHOULD ignore duplicate UDP initialization packets once it has replied with a notification.

8.4 Exchanging UDP Packets

Once the UDP association is established, UDP packets can be exchanged with the StreamHost. When a UDP packet is sent by either party, it MUST contain a 4-byte header (in addition to other possible headers, such as that of SOCKS5), which consists of the source virtual port and then the destination virtual port of the packet, both 16-bit values in network byte order. This allows the peers to multiplex many packets for different purposes over one session. The actual application data shall follow this header, and thus the payload size will always be "Application Data Size + 4".
For all packets sent to the StreamHost, DST.PORT is set to 0, and DATA contains the payload.

Listing 34: Sending UDP to StreamHost

```
ATYP = X'03'
DST.ADDR = SHA1 Hash of: (SID + Requester JID + Target JID)
DST.PORT = 0
DATA = (payload)
```

UDP packets sent from the StreamHost do not have any SOCKS5 headers, and so the payload shall be delivered as-is.
The programming interface for a SOCKS5 Bytestreams-aware UDP MUST report an available buffer space for UDP datagrams that is smaller than the actual space provided by the operating system and SOCKS5 layer if applicable. In other words, 4 more octets smaller.

9 Formal Description

9.1 <query/> Element

The <query/> element is the container for all in-band communications. This element MUST be qualified by the "http://jabber.org/protocol/bytestreams" namespace. Depending on the use case, this element contains multiple <streamhost/> elements, a single <streamhost-used/> element, or a single <activate/> element.
The 'sid' attribute specifies the bytestream session identifier. The value of this attribute is any character data. This attribute is REQUIRED.
The 'mode' attribute specifies the mode to use, either "tcp" or "udp". If this attribute is not included, the default value of "tcp" MUST be assumed. This attribute is OPTIONAL.
The 'dstaddr' attribute specifies the Requester's calculated value for the DST.ADDR field and is communicated from Requester to Target in certain situations (see Use with Multi-User Chat). This attribute is OPTIONAL.
The <streamhost/> element conveys the network connection information. At least one instance MUST be present in the initial IQ-set from the Requester to the Target. If multiple instances of this element are present, each one MUST be a separate host/port combination.
The <streamhost-used/> element informs the Requester about the StreamHost to which the Target has connected. It MUST be present in the IQ-set from the Target to the Requester, and there MUST be only one instance.
The <activate/> element is used to request activation of a unidirectional or bidirectional bytestream. It MUST be present in the IQ-set sent from the Requester to the Proxy after the Requester receives an IQ-result from the Target, and there MUST be only one instance.

9.2 <streamhost/> Element

The <streamhost/> element contains the bytestream connection information. This element has attributes for the StreamHost's JID, network host/address, and network port. This element MUST NOT contain any XML character data or child elements.
The "jid" attribute specifies the StreamHost's JID. This attribute MUST be present, and MUST be a valid JID for communication over XMPP.
The "host" attribute specifies the host to connect to. This attribute MUST be present. The value MUST be either an IPv4 or IPv6 address, or a resolvable DNS domain name.
The "port" attribute specifies the port to connect to. This attribute MAY be present. The value MUST be a valid port number in decimal form. If not specified, the port value is "1080" (in accordance with RFC 1928).
When communicating the available hosts, the Requester MUST include the host and port.

9.3 <streamhost-used/> Element

The <streamhost-used/> element informs the Requester about the StreamHost to which the Target has connected. This element has a single attribute for the JID of the StreamHost to

which the Target connected. This element MUST NOT contain any XML character data or child elements.
The ”jid” attribute specifies the JID of the StreamHost. This attribute MUST be present, and MUST be a valid JID for communication over XMPP.

9.4 <activate/> Element

The <activate/> element is sent from the Requester to the Proxy in order to formally start the bytestream. This element has no defined attributes and its XML character data specifies the JID of the target.

9.5 <udpsuccess/> Element

The <udpsuccess/> element is sent from the StreamHost to the Target or Requester to indicate that the StreamHost has received a UDP initialization packet.
This element is always empty and has one defined attribute, ”dstaddr”, which specifies the DST.ADDR that was used in the UDP datagram that the StreamHost received.

10 Implementation Notes

10.1 StreamHost Requirements

A StreamHost MUST support TCP connections.
A StreamHost SHOULD:

1. Allow bi-directional bytestreaming between the Requester and Target.
2. In the absence of explicit negotiation of multicasting with the Requester (methods for which are out of scope in this document), allow only one Target to connect to a bytestream.
3. Track sessions based on a combination of the StreamID and the Requester’s full or bare JID, thus allowing a Requester to create more than one simultaneous session.
4. Ignore any bytes sent before the bytestream is activated.

A StreamHost MAY:

1. Support UDP associations in addition TCP connections.
2. Ignore the DST.ADDR and DST.PORT parameters if desired.

10.2 SOCKS5 Parameter Mapping

To facilitate the usage of SOCKS5, command parameters MUST be mapped to the appropriate values. Parameters not specified in the table below SHOULD be used as defined in RFC 1928.

Parameter	Value
CMD	1 (CONNECT)
ATYP	Hardcoded to 3 (DOMAINNAME) in this usage
DST.ADDR	SHA1 Hash of: (SID + Requester JID + Target JID)
DST.PORT	0

Parameter	Value
CMD	3 (UDP ASSOCIATE)
ATYP	Hardcoded to 3 (DOMAINNAME) in this usage
DST.ADDR	SHA1 Hash of: (SID + Requester JID + Target JID)
DST.PORT	0

Parameter	Value
ATYP	Hardcoded to 3 (DOMAINNAME) in this usage
DST.ADDR	SHA1 Hash of: (SID + Requester JID + Target JID)
DST.PORT	0 or 1, for payload or initialization packets, respectively.

11 Security Considerations

11.1 Confidentiality and Integrity

This protocol does not include a method for securing or encrypting the data sent over a SOCKS5 bytetream. If such security is desired, it MUST be negotiated over the bytestream (once established) using standard protocols such as SSL or TLS. Negotiation of such security methods is outside the scope of this document.

11.2 Session Hijacking

In the absence of end-to-end encryption of the negotiation stanzas between the Requester and the Target, a passive attacker (eavesdropper) could authenticate to the bytestream before

the Target, thus preventing the Target from connecting and also hijacking the data sent from the Requester.

11.3 Denial of Service

A SOCKS5 Bytestreams Proxy can be subject to denial of service attacks (e.g., generating a large number of session requests that are never activated). Proxy deployments are advised to monitor usage from particular entities and blacklist them if their usage is excessive.

11.4 Use of SHA-1

The use of the SHA-1 algorithm to hash the SID, Requester's JID, and Target's JID is not security-critical. Therefore, the known weaknesses of SHA-1 are not of significant concern in this protocol.

12 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [17].

However, it is possible that a future version of this document will request assignment of a TCP/UDP port for SOCKS5 Bytestreams.

13 XMPP Registrar Considerations

13.1 Protocol Namespaces

The XMPP Registrar [18] includes 'http://jabber.org/protocol/bytestreams' in its registry of protocol namespaces.

13.2 Service Discovery Features

The XMPP Registrar includes 'http://jabber.org/protocol/bytestreams#udp' in its registry of service discovery features.

[17] 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/>.

[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/>.

13.3 Service Discovery Category/Type

The XMPP Registrar includes the ”proxy” category and associated ”bytestreams” type in the Service Discovery registry. The registry submission is as follows:

```
<category>
    <name>proxy</name>
    <desc>Proxy servers or services</desc>
    <type>
      <name>bytestreams</name>
      <desc>A proxy for SOCKS5 bytestreams</desc>
      <doc>XEP-0065</doc>
    </type>
  </category>
```

14 Schema

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='http://jabber.org/protocol/bytestreams'
    xmlns='http://jabber.org/protocol/bytestreams'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0065: http://www.xmpp.org/extensions/xep-0065.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='query'>
    <xs:complexType>
      <xs:choice>
        <xs:element ref='streamhost' minOccurs='0' maxOccurs='
            unbounded'/>
        <xs:element ref='streamhost-used' minOccurs='0'/>
        <xs:element name='activate' type='xs:string' minOccurs='0'/>
      </xs:choice>
      <xs:attribute name='dstaddr' type='xs:string' use='optional'/>
      <xs:attribute name='mode' use='optional' default='tcp'>
        <xs:simpleType>
          <xs:restriction base='xs:NCName'>
            <xs:enumeration value='tcp'/>
            <xs:enumeration value='udp'/>
          </xs:restriction>
```

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

  <xs:element name='streamhost'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='empty'>
          <xs:attribute name='jid' type='xs:string' use='required'/>
          <xs:attribute name='host' type='xs:string' use='required'/>
          <xs:attribute name='port' type='xs:string' use='optional'
              default='1080'/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name='streamhost-used'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='empty'>
          <xs:attribute name='jid' type='xs:string' use='required'/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name='udpsuccess'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='empty'>
          <xs:attribute name='dstaddr' type='xs:string' use='required'
              />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

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

</xs:schema>
```

15 Acknowledgements

Thanks to Marcus Lundblad, Henning Staib, and Matthew Wild for their feedback.