XEP-0003: Proxy Accept Socket Service (PASS)

Jeremie Miller
mailto:jer@jabber.org
xmpp:jer@jabber.org

Ryan Eatmon
mailto:reatmon@jabber.org
xmpp:reatmon@jabber.org

2009-06-03
Version 1.4

Status	Type	Short Name
Obsolete	Historical	pass

This document defines a method for relaying media via proxies on the Jabber/XMPP network.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction

Complete direct client-to-client file transfers presents a major problem for clients that are behind a firewall or NAT. Proxy Accept Socket Service (PASS) enables clients to do real-time file transfers via a third party; in addition, it does not limit clients to file transfers but enables any two clients to create a raw TCP socket between them for any purpose, such as VoIP (SIP/RTP), BEEP, or binary game data.

2 Registration/Setup

The first step is to communicate with a PASS service to set it up.

Listing 1: Registration request

```
<iq id='pass1' type='set' to='pass.jabber.org'>
  <query xmlns='jabber:iq:pass'>
    <expire>600</expire>
  </query>
</iq>
```

Listing 2: Result from PASS Service

```
<iq id='pass1' type='result' from='pass.jabber.org'>
  <query xmlns='jabber:iq:pass'>
    <server port='43253'>1.2.3.4</server>
  </query>
</iq>
```

At this point, the PASS service is now listening on the given IP and port for incoming connections on behalf of the Jabber Entity. The provided IP and port can now be sent to any other entity as a connection point, for file transfers or any other use.
The default behavior for the PASS service is to listen on the port forever, unless the requesting client specifies an <expire/> value (in seconds). If the service is not configured with an expire value, it will not timeout the connection, and will start listening on the port again after one of the two sides disconnects.

Code	Message	Cause
502	No more ports available. Try again later.	The PASS service is listening on all of its available ports.

3 Incoming Connections

When an incoming connection is attempted to that IP and port, the PASS service MUST send an IQ request to the entity on whose behalf it is listening:

Listing 3: Request to entity

```
<iq type='set'
    id='pass2'
    to='user@jabber.org/resource'
    from='pass.jabber.org'>
  <query xmlns='jabber:iq:pass'>
    <client port='1234'>4.3.2.1</client>
    <proxy port='43523'>1.2.3.4</proxy>
  </query>
</iq>
```

Listing 4: IQ result to accept connection

```
<iq type='result'
    id='pass2'
    from='user@jabber.org/resource'
    to='pass.jabber.org'>
  <query xmlns='jabber:iq:pass'/>
</iq>
```

The entity SHOULD now immediately connect to the given proxy IP and port, and upon connection all data written to that socket will be copied to the client connection, and vice-versa. Any disconnect on either side MUST cause a disconnect on the other (initiated by the service). If the IQ set to the entity fails or returns an error, the client socket MUST be dropped as well. The client XML element provides the information about the remote end of the incoming socket.
Abuse in bandwidth or resources could become an issue, so PASS implementations SHOULD include strict and detailed rate and usage limits, allowing only limited usage by single entities and rate-limiting bandwidth used if necessary for both single connections or overall usage. These limits are implementation-specific.

4 Controls

A Jabber client can send various controls to the PASS service via the set to control how the PASS service behaves, how the server handles a listening port.

4.1 expire

This tells the server to shut down the port after a specified number of seconds. After the timeout period, the PASS service MUST send an <iq/> to the JID to tell it that the port has been closed down. It notifies the client with:

Listing 5: Notification of expiration

```
<iq type='set'
    id='pass3'
    to='user@jabber.org/resource'
    from='pass.jabber.org'>
  <query xmlns='jabber:iq:pass'>
    <expire/>
    <close/>
    <proxy port='43253'>1.2.3.4</proxy>
  </query>
</iq>
```

Listing 6: Acknowledgement of expiration

```
<iq type='result'
    id='pass3'
    from='user@jabber.org/resource'
    to='pass.jabber.org'>
  <query xmlns='jabber:iq:pass'/>
</iq>
```

4.2 oneshot

This tells the server to listen once, and then close the port. Even if the <expire/> has not been met, the <oneshot/> overrides that and shuts down the listening port. When this happens the server notifies the client with the following packet:

Listing 7: Server notifies client of oneshot port closing

```
<iq type='set'
    id='pass4'
    to='user@jabber.org/resource'
    from='pass.jabber.org'>
  <query xmlns='jabber:iq:pass'>
    <oneshot/>
    <close/>
    <proxy port='43253'>1.2.3.4</proxy>
  </query>
</iq>
```

Listing 8: Client acknowledges closing of oneshot port

```
<iq type='result'
    id='pass4'
    from='user@jabber.org/resource'
    to='pass.jabber.org'>
  <query xmlns='jabber:iq:pass'/>
</iq>
```

4.3 close

This tells the server to explicitly shut down a listening port. Useful for when the client shuts down and can tell the PASS service to recycle the port. The server sends back:

Listing 9: Client request to shut down port

```
<iq type='set'
    id='pass5'
    to='pass.jabber.org'
    from='user@jabber.org/resource'>
  <query xmlns='jabber:iq:pass'>
    <close/>
    <proxy port='43253'>1.2.3.4</proxy>
  </query>
</iq>
```

Listing 10: Server acknowledges port closing request

```
<iq type='result'
    id='pass5'
    to='user@jabber.org/resource'
    from='pass.jabber.org'>
  <query xmlns='jabber:iq:pass'/>
</iq>
```

Code	Message	Cause
400	Missing <proxy/> specification.	Sent a <close/> w/o a <proxy/>
401	Port not registered to your JID.	You did not register this port with the server.
404	Port not found in registry.	The <proxy port=”/> was not a defined port.
405	Proxy IP does not match.	The IP sent in the <proxy/> does not match the IP of the pass-service

5 Security Considerations

5.1 Client Authentication

The PASS protocol can be used for clients to talk to each other and find out information about each other. When a client wants to send a file to another client, it can use the jabber:iq:pass namespace to query the IP address of the other client. For example:
You send the other client:

```
<iq type='get'
    id='pass6'
    to='them@jabber.org/resource'
    from='you@jabber.org/resource'>
  <query xmlns='jabber:iq:pass'/>
</iq>
```

The other client SHOULD send back:

```
<iq type='result'
    id='pass6'
    to='you@jabber.org/resource'
    from='them@jabber.org/resource'>
  <query xmlns='jabber:iq:pass'>
    <client>4.3.2.1</client>
  </query>
</iq>
```

Obviously the port is not going to be known, but the IP address will let you authenticate the JID via the PASS service since the PASS service tells you the <client/> information upon a connection.

5.2 Denying a Connection

When a server gets an Incoming Connection notification the client has the right to deny that connection based on the <client/> information that it receives. It can return an error to the PASS service specifying the <proxy/> port and hangup on the active <client/> connection and start listening again. This does not affect the <oneshot/> control. For example:
The PASS service sends you:

```
<iq type='set'
    id='pass7'
    to='user@jabber.org/resource'
    from='pass.jabber.org'>
  <query xmlns='jabber:iq:pass'>
    <client port='1234'>4.3.2.1</client>
```

```
    <proxy port='43523'>1.2.3.4</proxy>
  </query>
</iq>
```

Your client would send back:

```
<iq type='error'
    id='pass7'
    to='pass.jabber.org'
    from='user@jabber.org/resource'>
  <query xmlns='jabber:iq:pass'>
    <client port='1234'>4.3.2.1</client>
    <proxy port='43523'>1.2.3.4</proxy>
  </query>
  <error code='401' type='auth'>
    <not-authorized xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

Code	Message	Cause
401	Unauthorized	The incoming <client/> does not match the <client/> from the client you want to exchange data with.

6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [1].

7 XMPP Registrar Considerations

No action on the part of the XMPP Registrar [2] is necessary as a result of this document, since 'jabber:iq:pass' is already a registered protocol namespace.

[1] 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/>.

[2] 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/>.

8 XML Schema

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

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

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0003: http://www.xmpp.org/extensions/xep-0003.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='query'>
    <xs:complexType>
      <xs:choice minOccurs='0' maxOccurs='unbounded'>
        <xs:element name='client' type='PassEntity'/>
        <xs:element name='close' type='empty'/>
        <xs:element name='expire' type='xs:unsignedLong'/>
        <xs:element name='oneshot' type='empty'/>
        <xs:element name='proxy' type='PassEntity'/>
        <xs:element name='server' type='PassEntity'/>
      </xs:choice>
    </xs:complexType>
  </xs:element>

  <xs:complexType name='PassEntity'>
    <xs:simpleContent>
      <xs:extension base='xs:NMTOKEN'>
        <xs:attribute name='port' type='xs:short' use='optional'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

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

</xs:schema>
```