



XMPP

XEP-0046: DTCP

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

2003-04-11
Version 0.8

Status	Type	Short Name
Retracted	Standards Track	None

Direct TCP connection between two Jabber entities.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 - 2014 by the [XMPP Standards Foundation](#) (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the "Specification"), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

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 <http://xmpp.org/about-xmpp/xsf/xsf-ipr-policy/> or obtained by writing to XMPP Standards Foundation, 1899 Wynkoop Street, Suite 600, Denver, CO 80202 USA).

Contents

1	Introduction	1
2	Requirements	1
3	Implementation	1
3.1	Requesting a DTCP connection	1
3.2	Establishing the TCP connection	2
3.3	The handshake	3
3.3.1	SSL/TLS	3
3.3.2	Authenticating	3

1 Introduction

There are cases where it would be more optimal for clients to exchange data directly with each other rather than through the server. DTCP specifies a method for establishing a direct TCP socket connection between two entities, so that a reliable bytestream can be created out-of-band.

2 Requirements

The following design goals are considered:

- The protocol should be reasonably effective in scenarios involving NAT and/or firewalls.¹
- It should be reasonably secure.
- Establishing a connection should be fast.
- The protocol should be simple.

3 Implementation

3.1 Requesting a DTCP connection

Say you wish to initiate a DTCP session with Joe:

Listing 1: Requesting a DTCP session

```
<iq type="set" id="dtcp_1" to="joe@blow.com/Home">
  <query xmlns="http://jabber.org/protocol/dtcp">
    <key>c7b5ea3f</key>
    <host>192.168.0.32:8000</host>
    <host>63.110.44.12:8000</host>
  </query>
</iq>
```

The 'key' given is a unique key for Joe to use when referencing this session with you. If a 'host' element is present, then you are indicating that you can be reached at the given "host:port". Multiple hosts may be specified, but Joe cannot be expected to act on more than three of them.

¹DTCP works in situations where at least one client can accept incoming connections.

Listing 2: Success response

```
<iq type="result" id="dtcp_1" from="joe@blow.com/Home">
  <query xmlns="http://jabber.org/protocol/dtcp">
    <key>a1b2c3d4</key>
    <host>192.168.0.33:8000</host>
  </query>
</iq>
```

The success response is in exactly the same format as the request. As before, Joe cannot expect you to act on more than three hosts. The 'key' is a unique key from Joe that you will use when referring to the session with him.

Listing 3: Error response

```
<iq type="error" id="dtcp_1" from="joe@blow.com/Home">
  <error code="501">DTCP not supported</error>
</iq>
```

Or he may send an error.

3.2 Establishing the TCP connection

If you received a success response, then the next step is to form the TCP connection. Each entity should have a list of hosts (between 0-3 inclusive) and key of the other. With this information, they should each try to establish a direct connection with the hosts provided. When these connections take place is implementation dependent. Clients may choose to connect to all provided hosts at once, and both clients may even end up connecting to each other simultaneously. Clients may delay between connections, etc. As such, clients that are listening for connections should be prepared for anything.

The procedure ends when either a successful DTCP connection is formed (and all other TCP connections discarded), or when both entities have given up. An entity gives up when it is no longer trying to connect to any hosts. This is done by sending an additional iq-error packet, with the key of the other entity:

```
<iq type="error" to="joe@blow.com/Home">
  <query xmlns="http://jabber.org/protocol/dtcp">
    <key>a1b2c3d4</key>
  </query>
  <error code="503">Could not connect to any of the hosts.</error>
</iq>
```

If an entity was not provided any hosts, then it is assumed that he has given up and this packet need not be sent.

3.3 The handshake

For a given host, a TCP socket connection is established. Once connected, the connecting client must send a command across the channel. Each command is a line of text terminated by the ASCII linefeed character (0x0A).

Listing 4: Command format

```
[command]:[argument]<LF>
```

Some commands may have an argument, which is placed on the same line and separated by a colon character. If there is no argument, then the colon need not be present.

The serving client should keep the connection open after responding to a command, even if it resulted in an error, in case the connecting client wishes to try another command.

3.3.1 SSL/TLS

If you want an encrypted channel, then it must be requested using the 'starttls' command

```
starttls<LF>
```

If successful, the serving client should send back:

```
ok<LF>
```

This means that the serving client supports SSL and the connecting client should begin the SSL/TLS negotiation. After this, further data sent/received over the channel should be in encrypted form.

Or the serving client might report an error:

```
error<LF>
```

This means SSL is not supported by the serving client.

3.3.2 Authenticating

To complete the DTCP connection, the connecting client must authenticate with the serving client. This is done by exchanging keys. First, the connecting client sends the serving client's key:

```
key:a1b2c3d4<LF>
```

If the serving client recognizes the key, then it should respond by sending the connecting client's key:

```
ok:c7b5ea3f<LF>
```

Or the serving client might report an error:

```
error<LF>
```

On success, there may be one more step necessary. If the connecting client is also the original requestor of the DTCP connection (ie, he did the iq-set), then he must send the following "ack":

```
ok<LF>
```

This gives the final say to the requestor, to prevent any sort of race-condition due to the clients contacting each other at the same time. If the serving client is the requestor, then this extra "ack" is NOT needed and must NOT be sent.

At this point, the channel is established. No further commands may be issued, as the TCP connection is now for application data only.