

XMPP: basics, security, constrained networks (2023)

- **XMPP** (formerly known as **Jabber**) is an open standard approved by **IETF** (Internet Engineering Task Force)
 - IETF and **XSF** (XMPP Standards Foundation) maintain guidelines for XMPP
 - Other software such as Signal are good, but they're not open standards, so IETF approval doesn't apply to them
- Core features of XMPP are defined by **RFC**'s, which can be found at ietf.org
- Extended features of XMPP are defined by XEP's (XMPP Extension Protocols), which can be found at xmpp.org/extensions
- Aside from chatting, XMPP can be used for automation (via instructions), feeds and micro-blogging
- **Jingle** is for audio or video communication, which works on top of XMPP

Users

- 50 million XMPP users in 2009
- Zoom, Kontalk and Jitsi messaging services are based on XMPP
 - combined, these consist of over 200 million users
- There are over 1 billion users on XMPP based proprietary networks
- Google and DuckDuckGo used to offer XMPP services

User accounts

A Jabber Identifier (**JID**) consists of the username, and domain name of the XMPP user account. This can also include the extended resource part, which is an id for each device or connection.

IM Observatory is an XMPP server directory that tests server results by grade: <https://xmpp.net>. Servers not listed yet or that haven't been checked recently can be tested on this website. Server to server (**s2s**), and client to server (**c2s**) connections can be checked. This security grade goes by technical measures, so also, go by server's reputation.

To test XMPP servers for limited XEP's that are used with Conversations Messenger: <https://compliance.conversations.im/tests/>. This can also be used to find xmpp servers for use. <https://providers.xmpp.net/> is another server directory. For XMPP chatrooms, see <https://search.jabber.network/rooms/1>

You can get a user account (JID) at one of these servers listed in one of the directories in the websites listed above.

Opensource and Mobile Clients

All within the *net-im* directory of *Ports*:

kaidan – modern XMPP client available on FreeBSD which uses QT

dino – modern XMPP client available on FreeBSD which uses Vala and GTK

gajim – GTK+ Jabber client

kopete – KDE instant messenger

jxsc – browser client written in JavaScript, uses BOSH, rather than Websocket

pidgin – multiprotocol messenger program

To find more XMPP clients, utilities and accessories on FreeBSD, use: *psearch -c net-im -so xmpp jabber*

This can be used with the *grep -i* command to narrow findings to servers, console clients or other specific criteria. There are a few more XMPP/Jabber related ports within the *Port's* net directory.

blabber.im offers a mobile phone client based on conversations.im client.

Stream and Connection

An XMPP communication session is a stream over a TCP connection that lasts the length of the conversation. There are 2 streams established: 1 for each direction. The server and client each create their own XML XMPP document that is added onto as a conversation progresses. An XMPP server doesn't **poll** the client for new messages or updates. Rather than constantly waiting for messages, the server sends that message or holds that message until the intended receiving client comes on as soon as it's received.

The route an XMPP communication takes is a direct route from client, to server, if applicable to another server, then to client: this is called **direct federation**. There may be another connection on each end for converting XMPP to HTTP(S) use. (email uses indirect federation, which many indirect routes are taken between servers.)

In the traditional mode of XMPP communication, messages can get lost and aren't resent.

The resource part of a full JID can be used for further routing communication. Typically, only one device receives a message, which can be set by priority. It can be configured for many devices to receive a message.

The clients on their end close their XML/XMPP documents when a disruption in the stream is sensed.

Internal XML Components

When connection is started, XML begins as normal, with: `<?xml version="1.0"?>`

Under this is the `<stream:stream>` element, which doesn't close until the end of the connection. Then, after the opening `<stream:stream>` element, there's the `<stream:features>` self-closing element, which lists the features available for use in XMPP.

3 primary XML elements exchanged between server and client are called **stanzas**: **Message**, **IQ** and **PRESENCE**.

message stanza

Attribute type for message: error, normal, chat, groupchat, headline. Other attributes for message: to, from, id. Directs where message is sent in groupchat.

iq stanza

Is used for queries. In groupchat, can be used to change user permissions, or kick someone out of room. Can be used to reserve chatroom name.

presence stanza

Indicates users' availability. Extensions can indicate when user is typing, and another presence stanza is sent when there's a pause in typing. Join notification is indicated by this stanza. Tells when client leaves, unless the server to server connection goes down unexpectedly. Can indicate client's role (temporary status) or affiliation (persists across logins) in chatroom: visitor, participant, member, moderator, admin, owner.

For use over HTTPS

For use with HTTP endpoints, XMPP uses a client manager (**CM**) on those ends.

Websocket is the current way of doing this, and it's meant for use with HTML5. **BOSH** (Bidirectional-streams Over Synchronous HTTP) is the older way, which is incompatible with HTML5, but it's still widely used. Often, a CM will be at the same physical location as the XMPP server.

While Websocket had disadvantages over BOSH, especially on intermittent and mobile networks, modern XEP's have made up for those problems, so Websockets is the way to go. Otherwise, without those XEP's, Websockets has less overhead and is faster than BOSH. Websocket comes with XMPP through RFC 7395, and it works over TCP, allowing connections in both directions.

WebSocket uses less bytes for a message transfer than BOSH. Bosh uses long polling, and has "high transport overhead compared to XMPP's native binding to TCP."

WebSocket overcomes webpolling problems over HTTP. Supposedly, WebSocket uses under 10 bytes of overhead for small messages. BOSH may use over 150 bytes of each pair of request and response headers.

wss:// is the URI prefix for secure WebSocket. Data is secured under this transport, and not via frameworks under it. wss shouldn't be redirected to un-secure protocols like ws:// and http://. WebSocket was meant to work under HTTP port 443 and less secure port 80. It is adaptable to have its own ports in the future, and to be improved to have a better handshake. For details on how WebSocket functions: <https://blog.wildix.com/bosh-websocket-protocols/>.

Enabling Websocket on an XMPP server requires configuration of a module. Websocket uses polling to determine if messages were sent.

An XMPP server that uses Websocket or BOSH may have configuration information for use with an XMPP client. BOSH is obsolete, but is still in widespread use. The webpage of the Websocket module for Ejabberd describes Websocket as "a more elegant, modern and faster replacement to BOSH."

Setting up Websocket for client use

Check if your XMPP server has Websocket capability. See if website that hosts your XMPP server has a Websocket URL. For instance, <https://dismail.de/info.html>:

```
XMPP Server: ejabberd
s2s encryption: is required
Client-Port: 5222 (STARTTLS only)
MUC: rooms.dismail.de (multi user chat)
```

Proxy: proxy65.dismail.de Port 5555 (dismail.de only)
BOSH URL: https://dismail.de/http-bind
Websocket: wss://dismail.de/ws

Constrained and Intermittent Networks

Because XMPP uses streams that require a constant connection, it isn't suitable, as is, for mobile phones, intermittent Internet and other constrained networks. An intermittent Internet service causes the network to have to be reestablished, using up plenty of bandwidth. This is where Websocket and certain XEP's come into use. **Websocket** allows a **connection manager (CM)** to sit between the XMPP server and client, to take care of the XML end, and send (primarily stanza) data to the device, without it requiring a constant connection.

XMPP uses a direct federation model, so chose a suitable XMPP server and Client Manager (if applicable) relatively close to your geographical region. Server to server connections, are often much more reliable. A shorter distance from a client on a constrained network to a server, would be much better, than a client having to send messages a longer distance to another client on its own.

Compression protocols using zlib or lzw for XMPP have been outdated for some time. Methods of compression would go under <stream:features>, and they were meant to lower bandwidth of each stanza at a time. XEP-0322 (Efficient XML Interchange [EXI] Format) is the forerunner for XMPP stream compression: it is in deferred, but not obsolete status. EXI needs an update, because at this time, one listed option for using it relies on an obsolete XEP.

It is better that the server handles most of the features, so that the client won't have to do as much heavy lifting.

Stream Management (XEP-0198) synchronizes messages for intermittent networks (intended for mobile phones) when XMPP reconnects. This extension improves the use of Websocket.

XEP-0352: Client State Indication (CSI) is an extension that saves bandwidth for mobile phones. It reduces the need for establishing XMPP connections and resending data required for interrupted XMPP streams. It holds messages at the server, until the client comes online. This extension is available on some mobile clients like Conversations.

Presence settings can typically use around 90% of bandwidth data. For constrained networks, use minimal presence settings, or turn this off depending on needs. For more on mobile connections, see: Mobile Considerations on LTE Networks [[XEP-0286](#)]. See also: <https://www.websocket.org/aboutwebsocket.html>.

XEP-0365 is an interesting experimental extension, which is put into professional use by isode.

Severs that are preconfigured for security for use with each other can use Zero Handshake Server to Server Protocol (XEP-0361), which improves reliability in intermittent networks. Isode is a company that has an experimental extention for XMPP over low bandwidth or intermittent radio transmissions, which it implements in a professional capacity.

Privacy and Security

XMPP chats can be encrypted using OMEMO (XEP-0396) or PGP. For most uses, OMEMO is sufficient. OMEMO was adopted from Signal Messenger. PGP is supposedly more secure than OMEMO, but is difficult to set up. OTR was the old way of encrypting messaging, and it wasn't good for poor connections.

While messages to the servers are usually encrypted, the servers can still have access to this as unencrypted data for viewing and logging, unless end to end encryption such as OMEMO or PGP is used. When messages are end-to-end encrypted, metadata can still be seen by the server.

When OMEMO versions stop working with each other, OMEMO or its cryptographic dependencies may need to be updated for it to work again.

Direct spoofing is eliminated from the origins of the client end. The server adds relevant information to the *from* attribute from the client, rather than being added by the client itself. Similar characters substituted in Unicode text can be used in a client address in an attempt to spoof or deceive the reader. Spoofing is also eliminated from being originated between two secure servers, because the servers use a direct connection through a direct federation. Any attempt of direct spoofing would have to come from the server itself.

If a XMPP connection manager (CM) is used (this relates to Websocket and BOSH), that is another travel point that must be secured, unless it is on the same computer as the XMPP server.

It's important to choose a reputable server and if applicable client manager, that not only passes technical quality measures.

Certificate authority (CA) is available for free for XMPP servers through xmpp.org/ca. This is important to avoid the use of self-signed certificates, which can be a vulnerability for network spoofing. According to <https://blog.prosody.im/mandatory-encryption-on-xmpp-starts-today/>, there was intent within the XMPP community to replace CA's (Certificate Authorities) with models in the spirit of XMPP. DNSSEC, <http://web.monkeysphere.info/>, and fingerprints were mentioned.

XMPP's core comes with **STARTTLS** and **SASL** for securing connections. Before 2015, this was optional for XMPP software. It's recommended that administrators turn these features on for server and client connections. The connection starts off unencrypted until TLS negotiates and encryption is set. Still, XMPP requires further security measures.

Don't use **PLAIN** text for sending authentication/password data, unless the connection is already encrypted with TLS. Password is sent over base64. Once authenticated, a new XML session is started, with <stream:features> set with the enabled security features. Anonymous is an option for allowing clients to use an XMPP server without a password, for instance, when a service will be used very few times.

As for connecting, avoid **server dialback**, as this can be spoofed. DNS validation needs DNSSEC for better security.

Media

Jingle

Jingle is used for audio or video conferencing, and can be used for file transfer. A connection is established using XMPP, then Jingle uses a more direct connection to clients. Use ICE signaling which is designed for optimal use with NAT. Jingle also needs to be set up with OMEMO (XEP-0396: Jingle Encrypted Transports - OMEMO) to encrypt transport, as basic OMEMO for XMPP doesn't cover Jingle. Setting up, configuration and detailed use of Jingle is outside the scope of this document.

File transfer

SOCKS5 (XEP-0065) or HTTP File Upload (XEP-0363) are the main ways of transferring large files.

Jingle File Transfer (XEP-0234) is an alternate way to transfer large files, which it relies on the Jingle (XEP-0166) framework. SI File Transfer (XEP-0096) is obsolete.

For transferring smaller files over XMPP, Bits of Binary (XEP-0231) or In-Band Bytestreams (IBB: XEP-0047) can be used. These are inefficient for larger files, and can cause congestion. Bits of Binary's recommended file size limit is a maximum of 8KB. Aside from IBB's intended use for small files, it is also a fallback for Jingle File Transfer (XEP-0234). IBB can be a source of congestion on constrained networks. Bits of Binary and IBB encode the file into base 64, then break it down, to the maximum size allowed for each message to be transferred. It is sent by message, with an indication of sequence. The IQ stanza then returns a receipt, that the messages were delivered. Bits of Binary can be a security concern, because it sends binary data.

Additional uses of XMPP

These generally require extensions.

Remote commands

Can be used to forward and manage XMPP messages, and to control client devices. An additional use is for automated devices (automation data). IQ stanza is to query capabilities. Data can also be sent to SOAP and RPC protocols.

Pubsub

Publish/subscribe can be used for micro-blogging, and subscribing to other updates. Each subscription is a node on a Jabber ID (JID).

Data broadcasting and chat

MUC (Multi-user Chat) isn't only for chatrooms. MUC simply multiplies a message for many receivers. Data broadcasting is for relaying information to multiple devices.

Mediated Information eXchange (MIX), XEP-0369, is intended to be a replacement for MUC, for its perceived shortcomings. This extension is in the experimental state.

Other

- vcard extension - For showing JID profiles. Can be queried from server, unless the information is wanted from a specific device on the base JID.
- chat state notification - Turning this off, saves bandwidth.
- privacy lists and blocking - Shows as if user hasn't logged on. Messages can get dropped by server.
- receipts - Can be sent to sender of message, to know if important message was received.
- message archiving
- advanced message processing
- discovery protocol - Find services on servers. disco#items discovers entities, and disco#info discovers features on these entities.
- data forms
- captcha forms
- geolocation (XEP-0080)
- atom feeds (RFC 4287) - Since Atom is in XML, it can be used directly by XMPP.
- jukebox/radio (XEP-0118)
- application and gaming data
- Chat Markers (XEP-0333) - Sends an indication to the sender when a message has been read.

Keeping up to Date

Every year, XMPP Compliance Suites are published as an XEP (XMPP extension), which can be found at <https://xmpp.org/extensions/>. Often XEP's have become obsolete and superseded by another XEP. RFC's pertaining to XMPP are rarely replaced.

The latest stable compliance suite is XEP-0459 (<https://xmpp.org/extensions/xep-0459.html>) for 2022, while the latest proposed compliance suite is XEP-0479 (<https://xmpp.org/extensions/xep-0479.html>) for 2023.

<https://xmpp.org/extensions/xep-0286.html> is relevant for Mobile considerations on LTE Networks.

<https://compliance.conversations.im/tests/> is one standard of compliance tests as a suggestion for extension use, which are compatible with the conversations.im mobile browser.

About

This text can be found at <https://forums.freebsd.org/threads/xmpp-basics-security-constrained-networks.77220>. It's an update of that thread for 2023.

References; Further Reading

- XMPP: Definitive Guide - Peter Saint-Andre, Kevin Smith, Remko Troncon. O'Reilly (2009)
- xmpp.org
- ietf.org
- XMPP related [threads](#) on forums.freebsd.org
- Practical XMPP - Lloyd Watkins, David Koelle. Packt (2016)
- The Definite Guide to HTML5 WebSocket: [Chapter 4] Building Instant Messaging and Chat Over Websocket
- <https://takebackourtech.org/xmpp-comeback/>
- <https://blog.wildix.com/bosh-websocket-protocols/>
- *A Public Statement Regarding Ubiquitous Encryption on the XMPP Network*, by Peter Saint-Andre of xmpp.org (XSF) and jabber.org, <https://raw.githubusercontent.com/stpeter/manifesto/master/manifesto.txt> (2013-2014)
- <https://raw.githubusercontent.com/stpeter/manifesto/master/manifesto.txt>
- <https://apps.dtic.mil/dtic/tr/fulltext/u2/a534103.pdf>
- https://developer.ibm.com/tutorials/x-xmppintro/?mhsrc=ibmsearch_a&mhq=xmpp
- <https://jabber.markmail.org/>
- <https://github.com/superfeedr/ejabberd-websockets>
- <https://www.lob.com/blog/websocket-org-is-down-here-is-an-alternative>
- https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API