Extensible Messaging and Presence Protocol (XMPP)

Protocol Introduction and Overview

Sam Whited
XSF Editor / Council
JID: sam@samwhited.com

April 20, 2021
XMPP is a network protocol for exchanging data between two entities in near-real-time.
Standards

XMPP standardization managed by the IETF. Responsibility for extensions delegated to the XMPP Standards Foundation.

- **IETF**
  - RFC 6120: XMPP Core
  - RFC 6121: XMPP IM
  - RFC 7590: Use of TLS in XMPP
  - RFC 7622: XMPP Address Format
  - ...

- **XSF**
  - XEP-0045: Multi-User Chat
  - XEP-0198: Stream Management
  - XEP-0367: Message Attaching
  - ...
1999  XMPP created by the Jabber open-source community.

2002  IETF forms the XMPP WG.

2004  RFC 3920, RFC 3921, RFC 3922, and RFC 3923 approved.

2008  Cisco acquires Jabber, Inc.

2011  RFCs superseded by RFC 6120, RFC 6121, and RFC 6122.

2014  Websocket subprotocol RFC 7395 created.

2015  Address format superseded by RFC 7622, TLS updates in RFC 7590.
eXtensible Messaging and Presence Protocol
(from 10,000 feet)

- XML streams (not documents)
- Elements with payloads
  - Stanzas: message, presence, iq
  - Other: auth, compress, ...
- Minimal core spec with extensions defined in XEP’s
- Federated network of servers
Server 2

C1

S1

c2s

C2

S

C

C

C
XMPP Address Format

RFC 7622
Anatomy of a JID

viola@shakespeare.lit/ilyria

localpart  domainpart  resourcepart
Anatomy of a JID

viola@shakespeare.lit/ilyria

- localpart
- domainpart
- resourcepart
Anatomy of a JID

JID

violashaakespeare.lit/ilyria

localpart

domainpart

resourcepart
Anatomy of a JID

JID

viola@shakespeare.lit/ilyria

Bare JID

resourcepart
Anatomy of a JID

JID

viola@shakespeare.lit/ilyria

localpart

domainpart

resourcepart
XMPP Core

RFC 6120
Stream’s

- Client first
- Two streams, input and output, over one TCP socket
- As a security measure, streams are restarted when their state changes (eg. TLS or stream compression)
- Event based and pipelined (async communication)
Stream Initialization and Feature Negotiation
<?xml version='1.0'?>
<stream:stream ...

Client

<?xml version='1.0'?><stream:stream ...

Server

<?xml version='1.0'?><stream:stream ...

Client

```xml
<?xml version='1.0'?>
<stream:stream ...
```

Server

```xml
<?xml version='1.0'?>
<stream:stream ...
<stream:features>
  <starttls ...
    <required />
  </starttls>
</stream:features>
```
Client

<?xml version='1.0'?>
<stream:stream ...

<starttls .../>

Server

<?xml version='1.0'?>
<stream:stream ...

<stream:features>
  <starttls ...
    <required />
  </starttls>
</stream:features>
Client

<?xml version='1.0'?>
<stream:stream ...>
  <starttls .../>
</stream:stream>

Server

<?xml version='1.0'?>
<stream:stream ...>
  <stream:features>
    <starttls ...>
      <required />
    </starttls>
  </stream:features>
  <proceed .../>
</stream:stream>
Client

<?xml version='1.0'?>
<stream:stream ...

<starttls ...

Server

<?xml version='1.0'?>
<stream:stream ...

<stream:features>
 <starttls ...
 
  <required />
  </starttls>

 </stream:features>

<proceed ...

01000111011001010111010000100000011000100110000101100011011010110010
00000111010001101111001000000111011101101111011100100110101100101110
Client

<?xml version='1.0'?>
<stream:stream ...

Server

<?xml version='1.0'?>
<stream:stream ...

<stream:features>
  <starttls ...
    <required />
  </starttls>
</stream:features>

<starttls ...

01000111011001010111010000100000011000100110000101100011011010110010000111011101101111011100100110101100101110

<proceed ...

000001110100011011110010000000111011101101111011100100110101100101110

<stream:stream ...>
Client

```
<stream:stream ...>
```

Server

```
<html version='1.0'>
<stream:stream ...>
<stream:features>
  <starttls ...
    <required />
    </starttls>
  </stream:features>
<proceed ...

0100011101100101011001000100000110001001100010110001101101011011001001000011011101101111011100100110101100101110
<stream:stream ...>
```

```
<stream:stream ...>
```
Client

\(<\text{starttls \ldots}/>\)

010001110110010101110100001000000110001001100011011100100000011101110110111101110010011010110011011011010110010

0000011101000110111100100000011101110110111101110010011010110011011011010110010

\(<\text{stream:stream \ldots}>\)

Server

\(</\text{stream:features}>\)

\(<\text{proceed \ldots}/>\)

010001110110010101110100001000000110001001100011011100100000011101110110111101110010011010110011011011010110010

0000011101000110111100100000011101110110111101110010011010110011011011010110010

\(<\text{stream:stream \ldots}>\)

\(<\text{stream:features \ldots}>\)

\(<\text{mechanisms \ldots}>\)

\(<\text{mechanism}>\)

SCRAM-SHA-1

\)</mechanism>

\)</mechanisms>
Client

<starttls ...

0100011101100101011101000010000001100010001100010000001100010001101111001000000111011101101111011100100110101100101110

<stream:stream ...

<auth>

Server

</stream:features>

<proceed ...

0000011101000110111100101000000111011101101111011100100110101100101110

<stream:stream ...

<stream:features ...

<mechanisms ...

<mechanism>

SCRAM-SHA-1

</mechanism>

</mechanisms>

<auth>
Client

<starttls />

01000111011001010111010000100000011000100110000101100011011010110110010
00000111010011011110010000000111011101101111011100100110101100101110

<stream:stream />

Server

<proceed />

<stream:stream />

<stream:features />

<mechanisms />

<mechanism>
  SCRAM-SHA-1
</mechanism>

</mechanisms>

<auth>
...

...
Client

<proceed .../>
01000111011001010111010000100000011000100110001101100110110110110010
000001101000110111100100000001110111011011100110101101100101110
<stream:stream ...>

Server

<proceed .../>
<stream:stream ...>
<stream:features ...>
<mechanisms ...>
<mechanism>
SCRAM-SHA-1
</mechanism>
</mechanisms>
<auth>
...
...

<auth>
...
00000111010001101111001000000111011101101111011100100110101100101110
<stream:stream ...

<stream:stream ...
<stream:features ...
<mechanisms ...
<mechanism>
SCRAM-SHA-1
</mechanism>
</mechanisms>
<auth>
...
...
Inside the stream
Stanza /ˈstænzə/ (plural stanzas) n.

1 A unit of a poem, written or printed as a paragraph; equivalent to a verse.
2 (computing) An XML element which acts as basic unit of meaning in XMPP.
Stanza’s

The basic primitives of XMPP.

• <message/>
• <iq/>
• <presence/>

These are the only routable elements in an XMPP stream.
One-to-one

Fire and forget

No ack

Useful for anything that does not require a response (chats, alerts, logging, etc.)

```xml
<message id='262' type='chat'
    to='feste@shakespeare.lit/house'>
    <body>
        What's a drunken man like, fool?
    </body>
    <request xmlns='urn:xmpp:receipts'/>
    <thread>FNltztLMBQhQakHwcFd</thread>
</message>
```
<iq/> ("Information query")

- One-to-one
- Acked
- Optional at-least-once delivery

```xml
<iq from='capulet.lit'
    to='juliet@capulet.lit/balcony'
    id='s2c1' type='get'>
    <ping xmlns='urn:xmpp:ping'/>
</iq>

<iq to='capulet.lit'
   from='juliet@capulet.lit/balcony'
   id='s2c1' type='result'/>
```
• Directed (one-to-one) or broadcast (one-to-many)
• Advertises entity availability to the network
• Payload’s for broadcast can ride along (entity capabilities, status messages, etc.)

```
<presence id='aeg8y7pd'
from='prospero@shakespeare.lit/cell'>
  <status>
    Now my charms are all o' erthrown
  </status>
  <show>away</show>
</presence>
```
Namespacing

Stanza payloads are handled based on their XML namespace. By recent convention, namespaces are versioned URN's.

```xml
<message from='juliet@capulet.lit'
         to='romeo@montague.lit/orchard'
         type='headline' id='tfasd'>
         <result xmlns='urn:xmpp:mam:1' queryid='f27'
                id='5d398-28273-f7382'>
            ...
         </result>
</message>
```
“XMPP is Sacred”

—XEP-0134: XMPP Design Guidelines

When designing a new extension, think very hard about your life before you invent new stream level elements, and never modify core protocol.
Useful Extensions
“Extensions are XMPP’s greatest strength, and its greatest weakness.”

— Pretty much everyone
XEP-0280: Message Carbons

- Copies incoming messages to resources that would otherwise not have received the message.
- Copies outgoing messages to your other connected resources
- Current behavior not well defined for special messages (Typing notifications, read state markers, etc.)
- It’s simple and gets the job done
- One day might be replaced by...
XEP-0313: Message Archive Management (MAM)

- Stores incoming and outgoing messages on server
- New clients can access history
- Clients that have been offline can catch up
Myth: XMPP is bad on mobile

Turns out that XMPP is actually very good on mobile devices, both on battery and bandwidth.¹ Historically, mobile clients have been very bad.

¹Isode has deployed XMPP over 9600 bit/s SATCOM and STANAG 5066 HF radio
XEP-0352: Client State Indication (CSI)

Clients indicate when they become “inactive” (screen goes off, app loses focus, etc.) or “active” with some simple top-level elements. Server does what it wants with that data (eg. don’t send presence or typing notifications and start sending push notifications).

```xml
<active xmlns='urn:xmpp:csi:0'/>
<inactive xmlns='urn:xmpp:csi:0'/>
```
XEP-0268: Mobile Considerations

Attempts to tell you everything you need to know about not eating your users’ battery.

TL;DR — Implement CSI, and when you detect that something is already being sent/received: Send/receive as much data as you can at once so the radio can go back to sleep. Compression is also good.

Disclaimer: I wrote this one and it’s still in progress; I tried to do my research, but your mileage may vary.
XEP-0198: Stream Management

• Stream resumption (very fast reconnects)
• Stanza acknowledgements

Has some problems around “zombie state’s” where clients are offline, but the stream hasn’t timed out yet. Like most things, the answer is probably MAM.
Services

- Nintendo Switch notifications
- Warface
- WhatsApp
- Grindr
- Zoom
- Conversations.im
- Cisco Jabber
- Google Cloud Print (GCP)
- Firebase Cloud Messaging
- Jitsi Meet
Servers

- Prosody (Lua)
- Ejabberd (Erlang)
- MongooseIM (Erlang)
- M-Link (C++)
- Openfire (Java)
- Tigase (Java)
IM Clients

- Conversations (Android)
- Dino (Linux, MacOS, Windows)
- Swift (Linux, MacOS, Windows)
- Gajim (Linux, Windows)
- Apple Messages (MacOS)
- Siskin/Beagle (iOS, MacOS)
Libraries

- JVM (Java, Clojure, Scala, etc.)
  - Smack
  - Babble
- Python
  - aioxmpp
  - Words (Twisted)
- Lua: Verse
- Go: mellium.im/xmpp
- Rust: xmpp-rs
- JavaScript: Stanza.io
email & jid

sam@SamWhited.com

me

sr.ht & github

blog