XMPP Technologies

XMPP Technologies Overview

XMPP is the Extensible Messaging and Presence Protocol, a set of open technologies for instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data.

XMPP was originally developed in the Jabber open-source community to provide an open, secure, spam-free, decentralized alternative to the closed instant messaging services at that time. XMPP offers several key advantages over such services:

- **Open** — the XMPP protocols are free, open, public, and easily understandable; in addition, multiple implementations exist in the form clients, servers, server components, and code libraries.
- **Standard** — the Internet Engineering Task Force (IETF) has formalized the core XML streaming protocols as an approved instant messaging and presence technology. The XMPP specifications were published as RFC 3920 and RFC 3921 in 2004, and the XMPP Standards Foundation continues to publish many XEP series.
- **Proven** — the first Jabber/XMPP technologies were developed by Jeremie Miller in 1998 and are now quite stable; hundreds of developers are working on these technologies, there are tens of thousands of Jabber servers running on the Internet today, and millions of people use XMPP for instant messaging through public services such as Google Talk and XMPP deployments at organizations worldwide.
- **Decentralized** — the architecture of the XMPP network is similar to email; as a result, anyone can run their own XMPP server, enabling individuals and organizations to take control of their communications experience.
- **Secure** — any XMPP server may be isolated from the public network (e.g., on a company intranet), robust security using SASL and TLS has been built into the core XMPP specifications, and the XMPP network is virtually spam-free. In addition, the XMPP developer is actively working on end-to-end encryption to raise the security bar even further.
- **Extensible** — using the power of XML, anyone can build custom functionality on top of the core protocols; to maintain interoperability, common extensions are published in the XEP series, but such publication is not required and organizations can maintain their own private extensions if so desired.
- **Flexible** — XMPP applications beyond IM include network management, content syndication, collaboration tools, file sharing, gaming, remote systems monitoring, web services, lightweight middleware, cloud computing, and much more.
- **Diverse** — a wide range of companies and open-source projects use XMPP to build and deploy real-time applications and services; you will never get “locked in” when you use XMPP technologies.

The following pages provide an introduction to various XMPP technologies, including links to specifications, implementations, tutorials, and special-purpose discussion venues.

- Core — information about the core XMPP technologies for XML streaming
- BOSH — an HTTP binding for XMPP (and other) traffic
- Jingle — SIP-compatible multimedia signalling for voice, video, file transfer, and other applications
- Multi-User Chat — flexible, multi-party communication
- PubSub — alerts and notifications for data syndication, rich presence, and more

XMPP Technologies: Core

1. Overview

At its core, XMPP is a technology for streaming XML over a network. The protocol, which emerged from the Jabber open-source community in 1999, was originally designed to provide an open, secure, decentralized alternative to consumer-oriented instant messaging (IM) services like ICQ, AIM, and MSN. The core technologies were formalized under the name Extensible Messaging and Presence Protocol (XMPP) at the IETF in 2004. These core technologies include:

- The base XML streaming layer
- Channel encryption using Transport Layer Security (TLS)
- Strong authentication using the Simple Authentication and Security Layer (SASL)
- Use of UTF-8 for complete Unicode support, including fully internationalized addresses
- Built-in information about network availability ("presence")
- Presence subscriptions for two-way authorization
- Presence-enabled contact lists ("rosters")

2. Specifications

The core XMPP technologies are defined in two RFCs published by the IETF:

- RFC 3920: XMPP Core
- RFC 3921: XMPP IM

These RFCs are currently undergoing revision based on implementation and deployment experience since 2004. The revised versions are available at <http://xmpp.org/protocols/internet-drafts/>.

3. Implementations

There are many implementations of the core XMPP specifications. See the following pages for details:

- Clients
- Servers
- Code Libraries

XMPP Technologies: BOSH

1. Overview

BOSH is “Bidirectional-streams Over Synchronous HTTP”, a technology for two-way communication over the Hypertext Transfer Protocol (HTTP). BOSH emulates many of the transport primitives that are familiar from the Transmission Control Protocol (TCP). For applications that require both “push” and “pull” communications, BOSH is significantly more bandwidth-efficient and responsive than most other bidirectional HTTP-based transport protocols and the techniques known as AJAX. BOSH achieves this efficiency and low latency by avoiding HTTP polling, yet it does so without resorting to chunked HTTP responses as is done in the technique known as Comet. To date, BOSH has been used mainly as a transport for traffic exchanged between Jabber/XMPP clients and servers (e.g., to facilitate connections from web clients and from mobile clients on intermittent networks). However, BOSH is not tied solely to XMPP and can be used for other kinds of traffic, as well.

2. Specifications

BOSH is defined in two specifications:

- XEP-0124: Bidirectional-streams Over Synchronous HTTP
- XEP-0206: XMPP Over BOSH

3. Implementations

3.1 Servers

The following XMPP servers include built-in support for BOSH:

- ejabberd
- Jabber XCP
- Openfire
- Prosody
- Tigase

3.2 Connection Managers

The following standalone XMPP connection managers can be used with a wide variety of XMPP servers:

- Araneo
- JabberHTTPBind
- Punjab
- rhb

3.3 Clients

- Adium

- Gajim
- JWChat
- Pidgin
- Soashable
- SparkWeb
- Tigase Messenger
- Tigase Minichat

3.4 Libraries

- emite (gwt)
- gloox (C++)
- JSJaC (JavaScript)
- strophe (C or JavaScript)
- XIFF (Flash)
- XMPP4GWT (gwt)
- xmpp4js (JavaScript)
- XMPP4R (Ruby)

XMPP Technologies: Jingle

1. Overview

In essence, Jingle provides a way for Jabber clients to set up, manage, and tear down multimedia sessions. Such sessions can support a wide range of application types (such as voice chat, video chat, and file transfer) and use a wide range of media transport methods (such as TCP, UDP, RTP, or even in-band XMPP itself). The signalling to establish a Jingle session is sent over XMPP, and typically the media is sent directly peer-to-peer or through a media relay. Jingle provides a pluggable framework for both application types and media transports; in the case of voice and video chat, a Jingle negotiation usually results in use of the Real-time Transport Protocol (RTP) as the media transport and thus is compatible with existing multimedia technologies such as the Session Initiation Protocol (SIP). Furthermore, the semantics of Jingle signalling was designed to be consistent with both SIP and the Session Description Protocol (SDP), thus making it straightforward to provide signalling gateways between XMPP networks and SIP networks.

2. Specifications

Jingle is defined in a number of specifications:

- XEP-0166: Jingle
- XEP-0167: Jingle RTP Sessions
- XEP-0176: Jingle ICE-UDP Transport Method
- XEP-0177: Jingle Raw UDP Transport Method
- XEP-0181: Jingle DTMF
- XEP-0234: Jingle File Transfer

3. Implementations

Note: Many of the following implementations support the older Google Talk protocol and are being upgraded to support Jingle as it is defined in the specifications; contact the project developers for details.

3.1 Clients

- Coccinella
- Gajim
- Pandion
- Pidgin
- Psi
- SIP Communicator
- Telepathy
- Yate

3.2 Libraries

- libjingle (C/C++)
- Smack (Java)
- Telepathy Gabble (C)
- yjingle (C++)

3.3 Call Managers / VoIP Servers

- Asterisk
- FreeSWITCH
- Yate

XMPP Technologies: MUC

1. Overview

MUC is Multi-User Chat, an XMPP extension for multi-party information exchange similar to Internet Relay Chat (IRC), whereby multiple XMPP users can exchange messages in the context of a room or channel. In addition to standard chatroom features such as room topics and invitations, the protocol defines a strong room control model, including the ability to kick and ban users, to name room moderators and administrators, to require membership or passwords in order to join the room, etc. Because MUC rooms are based on XMPP, they can be used to exchange not only plaintext message bodies but a wide variety of XML payloads.

2. Specifications

MUC is defined in one primary specification (XEP-0045) and several ancillary specifications:

- XEP-0045: Multi-User Chat

- XEP-0249: Direct MUC Invitations
- XEP-0272: Multiparty Jingle

3. Implementations

3.1 Servers

The following XMPP servers include built-in support for MUC:

- ejabberd
- Jabber XCP
- M-Link
- Openfire
- Prosody
- Tigase

3.2 External Components

The following standalone components can be used with a wide variety of XMPP servers:

- mu-conference
- palaver

3.3 Clients

- Adium
- Gajim
- JWChat
- mcabber
- Pidgin
- Psi

3.4 Libraries

- AnyEvent:XMPP (Perl)
- gloox (C++)
- jabber-net (.Net)
- libpurple (C)
- Smack (Java)
- XMPP4R (Ruby)

XMPP Technologies: PubSub

1. Overview

XMPP PubSub is a protocol extension for generic publish-subscribe functionality, specified in XEP-0060. The protocol enables XMPP entities to create nodes (topics) at a pubsub service and publish information at those nodes; an event notification (with or without payload) is then broadcasted to all entities that have subscribed to the node. Pubsub therefore adheres to the classic Observer design pattern and can serve as the foundation for a wide variety of applications, including news feeds, content syndication, rich presence, geolocation, workflow systems, network management systems, and any other application that requires event notifications. The personal eventing protocol (PEP), specified in XEP-0163, provides a presence-aware profile of PubSub that enables every user's JabberID to function as a virtual pubsub service for rich presence, microblogging, social networking, and real-time interactions.

2. Specifications

PubSub is defined in several specifications:

- XEP-0060: Publish-Subscribe
- XEP-0163: Personal Eventing Protocol
- XEP-0248: PubSub Collection Nodes

2.1 Payloads

PubSub and PEP are "payload-agnostic" — you can use them as neutral transports for a wide variety of data formats. Some of the more popular payloads are listed below, especially for rich presence related to IM users:

- Activities
- Atom / RSS Notifications
- Avatars
- Chatroom Visits
- Gaming Activities
- Geolocation
- Moods
- Music Tunes
- TV/Video Activities
- Website Visits

3. Implementations

3.1 Servers

The following XMPP servers include built-in support for PubSub or PEP:

- ejabberd
- Jabber XCP
- Openfire
- Tigase

3.2 Server Components

- Idavoll

3.3 Clients

- Gajim

- Psi

3.4 Libraries

- strophe (C or JavaScript)
- XMPP4R (Ruby)