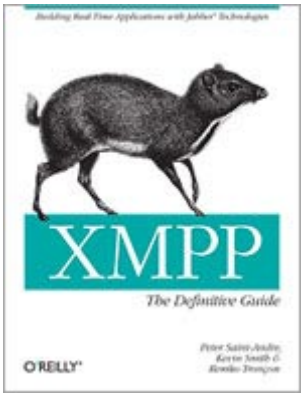
What Can You Do with XMPP?

By Kathryn Barrett
May 4, 2009 |



The following is an excerpt from the book **XMPP: The Definitive Guide** by **Peter Saint-Andre**, **Kevin Smith**, and **Remko Tronçon**. It has been adapted for the Web.

> *"XMPP is the secret weapon of the most groundbreaking websites, as well as the infrastructure behind hte fastest growing instant messaging systems. Not only is this sorely needed bible written by a dream-team of XMPP experts, the facts are interesting, the jokes are amusing, and quite honestly, I really wish I'd written it."*
> -Dave Cridland, XMPP Lead, Isode Ltd.

What Can You Do with XMPP?

The *Extensible Messaging and Presence Protocol* (XMPP) is an open technology for real-time communication, using the *Extensible Markup Language* (XML) as the base format for exchanging information. In essence, XMPP provides a way to send small pieces of XML from one entity to another in close to real time.

XMPP is used in a wide range of applications, and it may be right for your application, too. To envision the possibilities, it's helpful to break the XMPP universe down at a high level into *services* and *applications*. The services are defined in two primary specifications published by the Internet Engineering Task Force (IETF) at http://ietf.org/ (the "RFC" series), and in dozens of extension specifications published by the XMPP Standards Foundation at http://xmpp.org/ (the "XEP" series); the applications are software programs and deployment scenarios that are of common interest to individuals and organizations, although the core services enable you to build many other application types as well.

RFC Revisions

As of this writing, RFC3920 and RFC3921 are under active revision to incorporate errata, clarify ambiguities, improve their readability, define additional error codes, etc. These documents, called *rfc3920bis* and *rfc3921bis* in the terminology of the IETF,

provide the most accurate definition of XMPP and might have been published as replacement RFCs (with new numbers) once you read this book. For the latest versions of the revised specifications, visit http://xmpp.org.

Services

In this context, a *service* is a feature or function that can be used by any given application. XMPP implementations typically provide the following core services:

Channel encryption

This service, defined in RFC3920 and explained in Chapter 12 of this book, provides encryption of the connection between a client and a server, or between two servers. Although channel encryption is not necessarily exciting, it is an important building block for constructing secure applications.

Authentication

This service, also defined in RFC3920 and explained in Chapter 12 of this book, is another part of the foundation for secure application development. In this case, the authentication service ensures that entities attempting to communicate over the network are first authenticated by a server, which acts as a kind of gatekeeper for network access.

Presence

This service, defined in RFC3921 and explained in Chapter 3 of this book, enables you to find out about the network availability of other entities. At the most basic level, a presence service answers the question, "Is the entity online and available for communication, or offline and not available?" Presence data can also include more detailed information (such as whether a person is in a meeting). Typically, the sharing of presence information is based on an explicit presence subscription between two entities in order to protect the privacy of user information.

Contact lists

This service, also defined in RFC3921 and explained in Chapter 3 of this book, enables you to store a contact list, or *roster*, on an XMPP server. The most common use for this service is an instant messaging "friend list," but any entity that has an account on a server can use the service to maintain a list of known or trusted entities (e.g., it can be used by bots).

One-to-one messaging

This service, defined in RFC3920 and explained in Chapter 4 of this book, enables you to send messages to another entity. The classic use of one-to-one messaging is personal IM, but messages can be arbitrary XML, and any two entities on a network can exchange messages--they could be bots, servers, components, devices, XMPP-enabled web services, or any other XMPP entity.

Multi-party messaging

This service, defined in XEP-0045 and explained in Chapter 7 of this book, enables you to join a virtual chat room for the exchange of messages between multiple participants, similar to Internet Relay Chat (IRC). The messages can be plain text, or can contain XML extensions for more advanced functionality, such as room configuration, in-room voting, and various session control messages.

Notifications

This service, defined in XEP-0060 and explained in Chapter 8 of this book, enables you to generate a notification and have it delivered to multiple subscribers. This service is similar to multi-party messaging, but it is optimized for one-to-many delivery with explicit subscriptions to specific channels or topics (called "nodes").

Service discovery

This service, defined in XEP-0030 and explained in Chapter 5 of this book, enables you to find out which features are supported by another entity, as well as any additional entities that are associated with it (e.g., rooms hosted at a chat room service).

Capabilities advertisement

This service, defined in XEP-0115 and explained in Chapter 5 of this book, is an extension to the presence service that provides a shorthand notation for service discovery data so that you can easily cache the features that are supported by other entities on the network.

Structured data forms

This service, defined in XEP-0004 and explained in Chapter 6 of this book, enables you to exchange structured but flexible forms with other entities, similar to HTML forms. It is often used for configuration and other tasks where you need to gather ad-hoc information from other entities.

Workflow management

This service, defined in XEP-0050 and explained in Chapter 11 of this book, enables you to engage in a structured workflow interaction with another entity, with support for typical workflow actions, such as moving to the next stage of a business process or executing a command. It is often used in conjunction with data forms.

Peer-to-peer media sessions

This service, defined in XEP-0166 and explained in Chapter 9 of this book, enables you to negotiate and manage a media session with another entity. Such a session can be used for the purpose of voice chat, video chat, file transfer, and other real-time interactions.

These are some of the core services available to you (or your application) as a participant in an XMPP network. The XMPP developer community has defined additional features in various XMPP extensions, but here we focus on the services that we think you will find most useful in building real-time applications.

Applications

Given that you have a dozen core services at your disposal, what can you build? Here are a few possibilities:

Instant messaging

> The classic instantmessaging systems that most people are familiar with combine three of the core services: presence, contact lists, and one-to-one messaging. Such systems can and often do include more services and features, but if you have these three services, you can build a bare-bones IM application.

Groupchat

> The multi-party messaging service enables you to build groupchat systems similar to IRC. Often, groupchat systems are used for more specific applications, such as real-time trading systems in the financial industry, situation rooms for first responders and military personnel, and virtual classrooms.

Gaming

> Combined with custom extensions, both one-to-one messaging and multi-party messaging enable you to build simple gaming systems. For example, the Chesspark service (http://www.chesspark.com/) is built entirely using XMPP. Other game developers are using XMPP to add presence and contact list features to existing multi-party games.

Systems control

> The combination of one-to-one messaging and data forms makes it possible to deploy lightweight systems for control of and interaction with remote systems. Deployed applications in this domain include network management, scientific telemetry, and robotic control.

Geolocation

> The XMPP notification service is payload-agnostic. One defined payload format is geolocation, which enables you to build fascinating location-based applications, such as vehicle tracking.

Middleware and cloud computing

> A number of companies and research groups are actively working on XMPP-based systems for computation services, lightweight middleware, and management of cloud

computing infrastructures. While the use of XMPP may be surprising here because such applications have traditionally relied on heavyweight messaging technologies, we have seen XMPP begin to nibble away at the lower end of this market. It appears that companies that already have an XMPP infrastructure in place figure they might as well use it for non-IM use cases. These systems often use the workflow extensions we explore in Chapters Chapter 6 and Chapter 11 for structured message exchange. Specific applications include bioinformatics.

Data syndication

Popular social networking applications are increasingly using the XMPP notification service to solve a particular problem they have: constant polling for updated information. Existing HTTP-based deployments have been found not to scale, because quite often a particular feed has not changed since the last time it was polled. By contrast, the XMPP notification service sends out an update only when a feed has changed, saving a significant amount of bandwidth and server resources that otherwise would be wasted on polling.

Voice over IP (VoIP)

The Google Talk application that launched in August 2005 first popularized the use of XMPP for voice chat. Since then, the XMPP extensions for media session services (called Jingle) have been formalized through the XSF, and have been implemented and deployed by the likes of Nokia and the One Laptop Per Child project. The same extensions can also be used to negotiate a wide range of media session types, including video, file transfer, whiteboarding, and collaborative editing.

Identity services

Given the existence of stable identifiers (JabberIDs) and a robust authentication service, it is possible to use XMPP in building identity and authorization services such as OpenID and OAuth.

Other application examples include data transfer, live chat integrated into websites, mobile device communications, and presence-enabled directories. We will mention relevant applications throughout this book to illustrate the most popular and interesting uses of XMPP.

Although we highlight many applications of XMPP, unfortunately we can't cover all of them. Not only do we lack the space and time, but the list keeps growing every day. Moreover, the most cutting-edge uses of XMPP are not standardized yet, which makes them too much of a moving target to describe in a book. Examples of ongoing work at the time of this writing include collaborative document editing, whiteboarding, calendar integration, file sharing, and personal media networks. If you want to learn more about these topics, we suggest that you get involved with the XMPP community (see Chapter 13) as we define new ways of using XMPP.

What does the future hold for XMPP technologies? Although we don't know for sure, the trends seem clear: deployment of XMPP systems at more organizations and service providers, XMPP interfaces to more web applications, use of XMPP features to solve more business problems, and continued growth in the XMPP developer community. It's an exciting time to be working on XMPP technologies, and we invite you to join the conversation!

Brief History

Jabber/XMPP technologies were invented by Jeremie Miller in 1998. Jeremie was tired of running four different clients for the closed IM services of the day, so in true open source fashion, he decided to scratch an itch, releasing an open source server called *jabberd* on January 4, 1999. Before long, a community of developers jumped in to help, writing open source clients for Linux, Macintosh, and Windows; add-on components that worked with the server; and code libraries for languages such as Perl and Java. During 1999 and early 2000, the community collaboratively worked out the details of the wire protocols we now call XMPP, culminating in the release of *jabberd 1.0* in May 2000.

As the community grew larger and various companies became interested in building their own Jabber-compatible (but not necessarily open source) software, the loose collaboration evident in 1999 and 2000 became unsustainable. As a result, the community (spearheaded by a company called Jabber, Inc., acquired by Cisco in late 2008) formed the Jabber Software Foundation in August 2001. Ever since, this nonprofit membership organization, renamed the XMPP Standards Foundation in early 2007, has openly documented the protocols used in the developer community, and has defined a large number of extensions to the core protocols.

After several years of implementation and deployment experience, members of the developer community decided to seek a wider review of the core protocols by formalizing them within the IETF, which has standardized most of the core technologies for the Internet (including TCP/IP, HTTP, SMTP, POP, IMAP, and SSL/TLS). Given that most good protocols seem to be three- or four-letter acronyms ending with the letter "P," the relevant IETF working group labeled its topic the Extensible Messaging and Presence Protocol (XMPP). After less than two years of intensive work (mostly focused on tightening communications security), the IETF published the core XMPP specifications in its Request for Comments (RFC) series as RFC3920 and RFC3921 in October 2004.

Publication of these RFCs has resulted in widespread adoption of XMPP technologies. In August 2005, the Google Talk IM and Voice over Internet Protocol (VoIP) service was launched on a basis of XMPP. Thousands more services have followed. Prominent and emerging software companies use XMPP in their products, including the likes of Apple, Cisco, IBM, Nokia, and Sun. Countless businesses, universities, and government agencies have deployed XMPP-based instant messaging systems for their users. Many game developers and social networking applications are building XMPP into their services, and a number of organizations have used XMPP as the "secret sauce" behind some of their most innovative features.

Open Source and Open Standards

Although XMPP was originally developed in the Jabber open source community, the protocol itself is not an open source project like Apache, but rather an open standard like HTTP. As a result, XMPP is an open technology that is not tied to any single software project or company. The XMPP specifications define open protocols that are used for communication among network entities. Much as HTTP and HTML define the protocols and data formats that power the World Wide Web, XMPP defines the protocols and data formats that power real-time interactions over the Internet. The protocols are as free as the air, which means they can be implemented in code that is licensed as free software, open source software, shareware, freeware, commercial products, or in any other way. This open standards model is different from

the open source or free-software model for software code, wherein the code is often licensed so that modifications must be contributed back to the developers.

That said, XMPP emerged from an open source developer community, specifically the community that formed around the open source *jabberd* server that Jeremie Miller released. Thus there are many open source implementations of XMPP, which can be downloaded for free by end users, system administrators, and developers alike. Much of this software is focused on instant messaging, as befits a technology that started as an open alternative to closed IM silos that did not interoperate. There are open source clients for just about every operating system and device; as a result, millions of end users communicate using XMPP-based services. There are open source servers that can be deployed at companies, schools, and service providers; as a result, tens of thousands of XMPP services inter-connect every day. There are open source libraries for all the major programming languages, which can be used to write bots, components, and other real-time applications; as a result, there are thousands of active developers in the XMPP community. Much of this software is linked to from http://xmpp.org/, and we provide an overview of some of the most popular codebases in Appendix C.

Extensibility

The original Jabber developers were focused on building an instant messaging system. However, the extensible nature of XML has made XMPP attractive to application developers who need a reliable infrastructure for rapidly exchanging structured data, not just IM features. As a result, XMPP has been used to build a wide range of applications, including content syndication, alerts and notifications, lightweight middleware and web services, whiteboarding, multimedia session negotiation, intelligent workflows, geolocation, online gaming, social networking, and more.

Over the years, the developer community defined a large number of extensions to the core protocols. These extensions are developed through an open, collaborative standards process and published in the XSF's XMPP Extension Protocol (XEP) series at http://xmpp.org/. As you'll discover, the core protocols and various extensions provide a long "runway" for just about any feature you might need in developing real-time applications. But if you find that a feature is missing from the XMPP protocol stack, it is easy enough to extend the protocol for your own purpose, and (optionally) work with the community in standardizing these new features, as we discuss in Chapter 13.

Summary

In this excerpt from the book, we looked at the core services XMPP provides and sampled the kinds of applications you can build with those services. In the next sections, you'll get acquainted with the basic workings of XMPP, and then dive into each of the core XMPP services in detail. If you like what you've read here, get a copy of **XMPP: The Definitive Guide**