XEP-0134: XMPP Design Guidelines

Peter Saint-Andre
mailto:xsf@stpeter.im
xmpp:peter@jabber.org
http://stpeter.im/

2004-12-09
Version 1.1

Status	Type	Short Name
Active	Informational	N/A

This document defines best practices for the intelligent design of Jabber protocols and other XMPP extensions.

Legal

Copyright

Permissions

Warranty

Liability

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 <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction

The Extensible Messaging and Presence Protocol (XMPP) provides a solid, flexible foundation for a wide variety of applications on top of XMPP's core XML streaming technology. With the advancement of XMPP Core [1] and XMPP IM [2] within the Internet Standards Process, interest in building XMPP-based applications and extensions has accelerated even further. Unfortunately, not everyone who wants to build public or private XMPP extensions is familiar with the key design criteria that motivated the original developers of the Jabber technologies or that guide successful XMPP-based protocol design today. Thus there is value in attempting to translate the often-implicit knowledge held by long-time Jabber developers and protocol designers into more explicit policies and principles to which others can adhere. (For more general insights into Internet protocol design, see RFC 3117 [3].) The end result of explicating "The Jabber Way" will hopefully be a wider and deeper understanding of good protocol design practices within the Jabber/XMPP community.

2 Guidelines

2.1 XMPP is Sacred

Background
When the XMPP Standards Foundation (XSF) [4] submitted the XMPP Core and XMPP IM specifications to the Internet Engineering Task Force (IETF) [5], it ceded change control over the core XML streaming technology developed by the Jabber community. However, the XSF has reserved the right to define extensions to XMPP; furthermore, that right is not exclusive to the XSF, since anyone can define their own public or private extensions to XMPP. These extensions are usually in the form of structured XML data that is qualified by a unique namespace other than those currently reserved by the IETF or the XSF.
Meaning
When we say that "XMPP is Sacred", we mean that good protocol design must work within the context of XMPP and not require changes to the core protocols. For one thing, any such changes would need to be pursued within the IETF. Further, the core semantics most likely provide everything that a protocol designer needs. If you think that you need to define a new kind of top-level stanza (other than <message/>, <presence/>, and <iq/>) or a new value of the 'type' attribute for any stanza kind, then you need to think again. Treat XMPP as a transport

[1] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[2] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.
[3] RFC 3117: On the Design of Application Protocols <http://tools.ietf.org/html/rfc3117>.
[4] The XMPP Standards Foundation (XSF) is an independent, non-profit membership organization that develops open extensions to the IETF's Extensible Messaging and Presence Protocol (XMPP). For further information, see <https://xmpp.org/about/xmpp-standards-foundation>.
[5] The Internet Engineering Task Force is the principal body engaged in the development of new Internet standard specifications, best known for its work on standards such as HTTP and SMTP. For further information, see <http://www.ietf.org/>.

layer and build extensions on top of that layer (among other things, this implies that you must not modify the foundation when you are working on higher-level structures, for example by adding elements and attributes to the XMPP schemas on the theory that if applications will ignore them; define your own extensions in a separate namespace). A further implication of respecting XMPP is using structured data formats (e.g., applications of XML 1.0 [6] rather than binary or plaintext formats) whenever possible. Finally, as explained in XMPP Core, the <presence/> stanza exists to broadcast network and communications availability only; for more advanced information publishing, use Publish-Subscribe (XEP-0060) [7].

Examples

A good example of honoring the XMPP specifications is Invisibility (XEP-0126) [8]; while the Jabber community had informally defined <presence type='invisible'/> at one point, that protocol was abandoned in favor of an XMPP-compliant approach. Another example is XHTML-IM (XEP-0071) [9], which re-uses XHTML 1.0 [10] (a structured format that shares with XMPP a common root in XML) rather than Rich Text Format (RTF) [11] (an unstructured format that does not derive from XML). Further examples are the "extended presence" specifications (see Extended Presence Protocol Suite (XEP-0119) [12]), which are built on top of XEP-0060 rather than overloading the <presence/> stanza.

2.2 Keep Clients Simple

Background

Almost all Jabber technologies are implemented in a client-server architecture. While that's not necessary (and there do exist some peer-to-peer applications of XMPP), it usually makes good sense. Among other things, a client-server architecture has enabled the Jabber community to force most of the complexity onto servers and components, thus keeping clients relatively simple. This principle has served the Jabber community well since the very beginning, and forms an important basis for further innovation.

Meaning

The principle of "keep clients simple" has many implications, among them:

- Don't multiply protocols beyond necessity (the more protocols you define, the harder it is to write a client).
- Degrade gracefully so that simpler or older clients can still participate in the network.
- If intensive processing is required, make a server or component do it.

[6] Extensible Markup Language (XML) 1.0 (Fourth Edition) <http://www.w3.org/TR/REC-xml/>.
[7] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[8] XEP-0126: Invisibility <https://xmpp.org/extensions/xep-0126.html>.
[9] XEP-0071: XHTML-IM <https://xmpp.org/extensions/xep-0071.html>.
[10] XHTML 1.0 <http://www.w3.org/TR/xhtml1>.
[11] Rich Text Format (RTF) Version 1.5 Specification <http://msdn.microsoft.com/library/en-us/dnrtfspec/html/rtfspec.asp>.
[12] XEP-0119: Extended Presence Protocol Suite <https://xmpp.org/extensions/xep-0119.html>.

- Don't force additional requirements (such as XSLT processors) onto clients unless absolutely necessary.

Examples
One good example of keeping clients simple is the presence stanza: the client has only to send <presence/> and the server takes care of presence probes, broadcasts, and appropriate routing decisions. Another example is Multi-User Chat (XEP-0045) [13]: although the protocol involves some complexity, it was written so that older clients can join and participate in MUC rooms even if they don't understand the more advanced MUC extensions.

2.3 Re-Use Existing Protocols

Background
The Jabber community has been developing wire protocols for XML streaming, presence, and instant messaging since 1999. In that time, members of the community have defined a number of building blocks that can be used as the basis for further development. Furthermore, many smart people have created open protocols within other standards development organizations, including the IETF, the World Wide Web Consortium (W3C) [14], OASIS [15], the International Telecommunication Union (ITU) [16], and the Dublin Core Metadata Initiative (DCMI) [17].
Meaning
Good protocol designers "stand on the shoulders of giants" by re-using protocols that have been defined within the XSF and within other standards development organizations. That does not mean we don't define new protocols, because sometimes that is necessary. However, we are aware of work completed by others and we make use of it, especially when that work is outside the Jabber community's core competence areas (e.g., security or multimedia data formats rather than XML streaming, presence, and real-time messaging). Furthermore, the XSF prefers to re-use open protocols wherever possible. Finally, just as with XMPP, so also with XMPP extensions defined through the XSF: do not modify existing schemas (e.g., adding new elements and attributes) except through the XMPP extension process; instead, define extensions in a separate namespace).
Examples
Examples of re-using existing Jabber protocols include Stream Initiation (XEP-0095) [18] (which re-uses Feature Negotiation (XEP-0020) [19]) and XEP-0126: Invisibility (which re-uses the privacy lists protocol defined in XMPP IM). Examples of re-using non-Jabber protocols include

[13] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[14] The World Wide Web Consortium defines data formats and markup languages (such as HTML and XML) for use over the Internet. For further information, see <http://www.w3.org/>.
[15] OASIS is a not-for-profit, international consortium that drives the development, convergence and adoption of e-business standards. For further information, see <http://www.oasis-open.org/>.
[16] The International Telecommunication Union develops technical and operating standards (such as H.323) for international telecommunication services. For further information, see <http://www.itu.int/>.
[17] The Dublin Core Metadata Initiative (DCMI) is an organization dedicated to promoting the widespread adoption of interoperable metadata standards. For further information, see <http://www.dublincore.org/>.
[18] XEP-0095: Stream Initiation <https://xmpp.org/extensions/xep-0095.html>.
[19] XEP-0020: Feature Negotiation <https://xmpp.org/extensions/xep-0020.html>.

SOCKS5 Bytestreams (XEP-0065) [20] (which makes use of RFC 1928 [21]) and Common Alerting Protocol (CAP) over XMPP (XEP-0127) [22] (which defines a way to send Common Alerting Protocol [23] data via Jabber). Here again XEP-0071 provides an example: it re-uses XHTML 1.0 (an open protocol developed by a recognized standards development organization) rather than RTF (a closed protocol under the control of the Microsoft Corporation).

2.4 Modular is Better

Background
Most Jabber implementations are built using modular architectures, wherein pieces of functionality are coded as separate components and then assembled into larger wholes, with core routing logic that integrates the system (examples include clients that enable the development of plugins and servers that enable the attachment of external components). We can view many Jabber protocols the same way: each one specifies a well-defined domain of functionality that is loosely connected to other domains and integrated by the core transport layer provided by XMPP.
Meaning
The best Jabber protocols are quite focused and provide limited but powerful functionality that can be applied in a specific domain or, sometimes, re-used by other Jabber protocols. Even if the domain is more complex, a protocol that addresses it needs to clearly define its scope, limit that scope as much as possible, and specify only the protocols necessary to meet the core requirements.
Examples
Service Discovery (XEP-0030) [24] and Data Forms (XEP-0004) [25] are good examples of focused, single-purpose protocols. By contrast, Multi-User Chat is more complex, but it limits itself to the domain of text conferencing in the context of virtual rooms (e.g., it does not address service-level administration) and consists of separate namespaces for end-user, moderator, and room owner functionality. A good example of a protocol that is focused on a smaller domain is Roster Item Exchange (XEP-0144) [26].

2.5 Know Your Strengths

Background
The core strength of Jabber technologies is the streaming of relatively small XML fragments between presence-aware network endpoints. As is usually the case, our greatest strength

[20] XEP-0065: SOCKS5 Bytestreams <https://xmpp.org/extensions/xep-0065.html>.
[21] RFC 1928: SOCKS Protocol Version 5 <http://tools.ietf.org/html/rfc1928>.
[22] XEP-0127: Common Alerting Protocol (CAP) over XMPP <https://xmpp.org/extensions/xep-0127.html>.
[23] Common Alerting Protocol, v. 1.0 <http://www.oasis-open.org/committees/documents.php?wg_abbrev=emergency>.
[24] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[25] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.
[26] XEP-0144: Roster Item Exchange <https://xmpp.org/extensions/xep-0144.html>.

is also our greatest weakness. Thus XMPP is not optimized for binary data, large XML files, multimedia streaming, or other such applications.

Meaning

It's not a bad thing that we don't solve the problems of exchanging binary data, streaming multimedia, or transferring large XML files, because other protocols and technologies have addressed those domains. But it's important to recognize what we do well and what we don't. For example, sending base64-encoded binary data, streaming voice or video, or consistently large stanzas in the Jabber band [27] is probably not a good idea, and applications that would depend on such behavior are better designed to communicate their data out of band.

Examples

SI File Transfer (XEP-0096) [28] is a good example of respecting the strengths and weaknesses of XMPP, since it specifies that going out of band is the preferred mechanism for bandwidth-heavy data transfers.

2.6 Be Explicit

Background

In the beginning was the code (mainly jabberd [29]). Although code is explicit in its own way, not everyone reads code, and detailed specifications are necessary in order to make functionality reproducible in different codebases. The Jabber community has learned that lesson the hard way.

Meaning

Detailed, explicit specifications are good specifications. Define your terms. Use conformance terminology such as MUST and SHOULD rather than loose English words such as "does" and "will". Follow the Guidelines for Authors of XMPP Extension Protocols (XEP-0143) [30]. Specify error conditions. Include lots of examples. Restrict the allowable XML via schemas and datatypes as specified in XML Schema Part 1 [31] and XML Schema Part 2 [32].

Examples

[27] There are no hard-and-fast rules regarding a reasonable upper limit on the average XML stanza. (Note the use of both 'reasonable' and 'average' in that sentence.) In reality, there is a continuum of stanza sizes, and different sizes may be appropriate for different types of XMPP applications and deployments. While sending 2 gigabyte or 2 megabyte stanzas is wrong in the current context of Jabber technologies, we cannot legitimately say that a 2 kilobyte, 20 kilobyte, or even 200 kilobyte stanza is unreasonable. Is the stanza sent over an open network with current server implementations, or over a closed network with specially tuned servers and clients? Does the application generate one such stanza every second, every minute, every hour? Considerations of this kind help us determine if the use of XMPP is "reasonable" in some sense. However, when protocol extensions are defined in XMPP Extension Protocols, the XMPP Council will require clear explanation of design choices and reasonable stanza size limits if the extension will generally require what might be considered larger than normal stanzas.

[28] XEP-0096: SI File Transfer <https://xmpp.org/extensions/xep-0096.html>.

[29] The jabberd server is the original server implementation of the Jabber/XMPP protocols, first developed by Jeremie Miller, inventor of Jabber. For further information, see <http://jabberd.org/>.

[30] XEP-0143: Guidelines for Authors of XMPP Extension Protocols <https://xmpp.org/extensions/xep-0143.html>.

[31] XML Schema Part 1: Structures <http://www.w3.org/TR/xmlschema11-1/>.

[32] XML Schema Part 2: Datatypes <http://www.w3.org/TR/xmlschema11-2/>.

XMPP Core and XMPP IM are large documents that define the Extensible Messaging and Presence Protocol in excruciating detail. Although such specifications are not fun to write, they provide a model for good protocol design and documentation.

2.7 Stay Flexible

Background
The need for explicit definition must be balanced against the need for flexibility. A completely rigid protocol may break under stress or when conditions change, whereas a more flexible protocol may bend and adapt. Knowing when to be explicit and when to be flexible is a key to good protocol design.
Meaning
In general, a protocol needs to define the skeleton of functionality, but not necessarily specific parameters or values to be used within a certain domain. In order to allow for growth and change, it often makes sense to specify that the XMPP Registrar [33] shall keep track of certain parameters and values, rather than to explicitly limit them in the protocol itself.
Examples
Whereas the old Agent Information (XEP-0094) [34] and Jabber Browsing (XEP-0011) [35] protocols defined certain hardcoded values for entity types and categories, Service Discovery has left that function up to the XMPP Registrar. Similarly, Stream Initiation (XEP-0095) [36] defines a registry for its profiles, Advanced Message Processing (XEP-0079) [37] defines registries for processing conditions and actions, and a number of XMPP Extension Protocols register FORM_TYPE values as specified in Field Standardization for Data Forms (XEP-0068) [38].

2.8 Privacy and Security Matter

Background
Since the beginning, privacy and security have been important priorities within the Jabber community. That must not change.
Meaning
Good protocols respect the confidentiality of data generated or communicated by users and applications as well as the security of the system or network as a whole. Although privacy and security considerations have been dealt with at the core XMPP layer, application-level protocols must not compromise privacy and security. Attention to these matters, along with rigorous cross-area review and close scrutiny by protocol designers (in the form of the XMPP

[33] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.
[34] XEP-0094: Agent Information <https://xmpp.org/extensions/xep-0094.html>.
[35] XEP-0011: Jabber Browsing <https://xmpp.org/extensions/xep-0011.html>.
[36] XEP-0095: Stream Initiation <https://xmpp.org/extensions/xep-0095.html>.
[37] XEP-0079: Advanced Message Processing <https://xmpp.org/extensions/xep-0079.html>.
[38] XEP-0068: Field Data Standardization for Data Forms <https://xmpp.org/extensions/xep-0068.html>.

Council [39] and Standards SIG [40]), will help ensure that the protocols we develop will provide a strong foundation for communication over the Internet.

Examples

As is well-known, the presence subscription model developed by the Jabber community and specified in XMPP IM requires approval before a contact can view a user's presence. Similarly, Jabber has always included strong authentication methods, which have been further improved through the use of SASL (RFC 4422 [41]).

3 Security Considerations

There are no security features or concerns directly related to this proposal, which is informational in nature. However, as discussed above, protocols that are developed following these guidelines should appropriately address privacy and security considerations. Helpful guidelines for security in relation to Internet protocol design can be found in RFC 3552 [42].

4 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [43].

5 XMPP Registrar Considerations

This document requires no interaction with the XMPP Registrar.

[39] The XMPP Council is a technical steering committee, authorized by the XSF Board of Directors and elected by XSF members, that approves of new XMPP Extensions Protocols and oversees the XSF's standards process. For further information, see <https://xmpp.org/about/xmpp-standards-foundation#council>.

[40] The Standards SIG is a standing Special Interest Group devoted to development of XMPP Extension Protocols. The discussion list of the Standards SIG is the primary venue for discussion of XMPP protocol extensions, as well as for announcements by the XMPP Extensions Editor and XMPP Registrar. To subscribe to the list or view the list archives, visit <https://mail.jabber.org/mailman/listinfo/standards/>.

[41] RFC 4422: Simple Authentication and Security Layer (SASL) <http://tools.ietf.org/html/rfc4422>.

[42] RFC 3552: Guidelines for Writing RFC Text on Security Considerations <http://tools.ietf.org/html/rfc3552>.

[43] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.