XEP-0143: Guidelines for Authors of XMPP Extension Protocols

Peter Saint-Andre
mailto:stpeter@stpeter.im
xmpp:stpeter@jabber.org
https://stpeter.im/

2016-12-02
Version 1.1.2

Status	Type	Short Name
Active	Procedural	N/A

This document provides information intended to assist authors of XMPP Extension Protocols.

Contents

1 Introduction

The XMPP Standards Foundation (XSF) [1] receives a significant number of proposals for defining extensions to the core XMPP protocols specified in XMPP Core [2]. However, it is not always clear to authors how to best structure a proposal in order for it to be accepted as an XMPP Extension Protocol (XEP) and then advance through the XSF's standards process. Therefore, this document provides guidelines that are intended to help authors write better XMPP Extension Protocol specifications.
These guidelines assume that the reader is familiar with the XEP series of documents and the processes for handling them within the XSF, as defined in XMPP Extension Protocols (XEP-0001) [3].

2 Before You Submit a Proposal

A prospective author is strongly encouraged to complete some research before submitting a proposal for consideration as a XEP. In particular, the author should do the following:

- Review the XMPP RFCs and Experimental, Active, Draft, and Final XEPs to determine if the proposed protocol extension is truly needed in order to fill a gap in existing XMPP technologies and protocols.
- Review rejected and deferred XEPs, as well as proposals that were never accepted (see <http://xmpp.org/extensions/inbox/>) to determine if similar extensions have been proposed in the past but not approved by the XMPP Council [4].
- Review protocols developed within other standards development organizations, such as the Internet Engineering Task Force (IETF) [5] and World Wide Web Consortium (W3C) [6], to determine if they might be more appropriate than a new XMPP extension.
- Review discussions within the Standards SIG [7] to determine if similar functionality has been discussed in the past or is currently under discussion.

[1]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>.
[2]RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[3]XEP-0001: XMPP Extension Protocols <https://xmpp.org/extensions/xep-0001.html>.
[4]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>.
[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/>.
[6]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/>.
[7]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

After completing this research, the prospective author might conclude that a new protocol extension is needed. If so, the author is strongly advised to do the following:

1. Review XMPP Extension Protocols (XEP-0001) [8] and the XMPP Design Guidelines (XEP-0134) [9].
2. Understand the Submission Process.
3. Become familiar with the XEP XML Format.
4. Then and only then write a proposal that includes all of the appropriate Sections of a XEP.
5. Review the content to ensure that it conforms to the XEP Styleguide.

3 Submitting a Proposal

The process for submitting a proposal for consideration as a XEP is straightforward:

1. Write your proposal following the guidelines described in this document.
2. Make sure that you read, understand, and agree to the XSF IPR Policy [10] before you submit your proposal.
3. Send your XML file (or a URL pointing to the file) to the XMPP Extensions Editor [11] team via email to editor@xmpp.org (make sure that the email subject includes the string "XEP" somewhere in it!)..

4 Maintaining a XEP

If your proposal is accepted as a XEP, you will probably need to update the specification periodically to incorporate feedback as well as implementation and deployment experience. The XMPP Extensions Editor team will assign a XEP number to your document and add it to source control.
The XMPP Extensions Editor team prefers that you work as follows:

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/>.

[8]XEP-0001: XMPP Extension Protocols <https://xmpp.org/extensions/xep-0001.html>.

[9]XEP-0134: XMPP Design Guidelines <https://xmpp.org/extensions/xep-0134.html>.

[10]The XSF IPR Policy defines the XMPP Standards Foundation's official policy regarding intellectual property rights (IPR) as they pertain to XMPP Extension Protocols (XEPs). For further information, see <https://xmpp.org/about/xsf/ipr-policy>.

[11]The XMPP Extensions Editor is the individual appointed by the XSF Board of Directors to handle protocol submissions and provide day-to-day management of the XSF's standards process. For further information, see <https://xmpp.org/about/xsf/editor-team>.

1. Create your own fork of the XSF's git repository (the address of the GitHub mirror can be found at <http://xmpp.org/about-xmpp/xsf/xsf-source-control/>).
2. Create a dedicated branch for these changes in your forked repository.
3. Make your desired changes to the document or documents, including an updated <revision/> element as described below.
4. Commit and push the changes to your branch.
5. Create a pull request from your fork to the master branch at the GitHub mirror.

The XMPP Extensions Editor team will then process your pull request, seek Council approval if necessary, and publish an updated version of your XEP.
NOTE: As explained in XMPP Extension Protocols (XEP-0001) [12], updated versions of XEPs in the Experimental state are published without the need for approval by the XMPP Council. However, updated versions of XEPs in the Active, Draft, or Final state must be approved by the XMPP Council to ensure proper change control regarding approved protocols.

5 XEP XML Format

The XEP XML format is substantially similar to a reduced set of XHTML. This is intentional: it makes it easier to author XEPs. In fact, if you use the template file with its associated XSLT stylesheet, you should be able to view your proposal in most modern web browsers (see below). The following subsections explain how to get started with XEP authoring and describe the XML format used for XEPs (see the xep.xsd or xep.dtd file for a formal description).

5.1 Working with XEP Files

The best way to start working on your proposal is to retrieve all of the existing XEP files and associated stylesheets from source control. These files are stored using the git system as described at <http://xmpp.org/about-xmpp/xsf/xsf-source-control/>. The document structure is formally defined by both a DTD and an XML schema, but you do not need to understand the formal descriptions in order to author a XEP. In addition, a handy template file is included as the 'xep-template.xml' file in the 'extensions' directory, providing a quick starting point for XEP authoring.
To create your proposal, do a git clone of the 'xeps' repository, go to the 'xeps/' directory you just cloned (e.g., 'cd xeps'), copy the template file (e.g., 'cp xep-template.xml xep-foo.xml'), and start editing the file using either a basic text editor or a specialized XML editing application such as XML Spy or XMLmind.
Even if you use a basic text editor, you should be able to view your document in most modern web browsers as an XML file as long as you have xep.xsl and xep.dtd in the 'xeps' directory.

[12] XEP-0001: XMPP Extension Protocols <https://xmpp.org/extensions/xep-0001.html>.

Because of inconsistencies in browser XSLT implementations, certain formatting (e.g., table layouts and the numbering of tables, examples, and footnotes) might not be perfect. Don't panic; it will look fine in the HTML output produced by the XMPP Extensions Editor team. If your XML file doesn't render at all (i.e., it's just one big text blob), you are using a bad browser. If you see only the bare outline generated by xep.xsl but none of your text, you have an error in your XML. You can check your XML syntax at xml.com [13].
To programatically convert your XML file into HTML, we recommend using Daniel Veillard's xsltproc program, which will give you helpful error messages regarding XML syntax problems. However, the XMPP Extensions Editor team will complete the final rendering of XML into HTML as well as posting of your HTML file to www.xmpp.org, so you do not need to generate HTML files for submission to the XMPP Extensions Editor team (in fact, the XMPP Extensions Editor team requires that you submit your proposal in the XEP XML format, not HTML).
Finally, the xep.ent file contains convenient "external entities" that provide shortcuts for including references to XMPP Extension Protocols, RFCs, and other common strings. Unfortunately, most browsers do not correctly process external entities, so you cannot include entities from xep.ent if you need to view your XML source file in a browser. However, the XMPP Extensions Editor team reserves the right to convert your markup to external entities, since it makes their life easier. Also, please do not add items to the xep.ent file; instead, add them as inline entities within your document and then ask the XMPP Extensions Editor team to add them to the xep.ent file.

5.2 File Metadata

This section describes the metadata elements contained in the <header/> element of a XEP file (see below for the file contents).
The XML character data of the <title/> element is the title of your XEP. Choose a descriptive title that is less than ten words long. The XMPP Extensions Editor team may change this in consultation with the author.
The XML character data of the <abstract/> element SHOULD be one or two sentences that capture the essence of your proposal (usually beginning "This specification defines an XMPP protocol extension that..."). The XMPP Extensions Editor team has been known to modify the abstract so that it accurately describes the proposal.
The XML character data of the <legal/> element MUST be as defined in the XSF IPR Policy and reflected in both the xep.ent file and the XEP template.
The XML character data of the <number/> element SHOULD be "xxxx"; this will be changed to the next sequential XEP number by the XMPP Extensions Editor team if the XMPP Council accepts the proposal as an XMPP Extension Protocol.
The XML character data of the <status/> element SHOULD be "ProtoXEP" since all proposals start out as "proto-XEPs"; this will be changed to "Experimental" if the XMPP Council accepts the proposal as an XMPP Extension Protocol.
The XML character data of the <type/> element SHOULD be either "Standards Track" or

[13]<http://www.xml.com/pub/a/tools/ruwf/check.html>

"Informational" (there are also Historical, Humorous, and Procedural XEPs, but these are uncommon and usually written by the XMPP Extensions Editor team). A Standards Track XEP defines an XMPP extension intended to be used as a common part of XMPP technologies. An Informational XEP defines best practices or a usage profile related to XMPP or an XMPP Extension Protocol (e.g., Best Practices for Use of SASL ANONYMOUS (XEP-0175) [14]).
The XML character data of the <approver/> element SHOULD be "Council".
The <dependencies/> element is used to specify RFCs, XMPP Extension Protocols, and other specifications on which your proposal depends in a normative fashion (i.e., specifications that MUST or SHOULD be understood in order to implement your proposed protocol). Each specification MUST be identified by a distinct <spec/> child element (see existing XEP specifications for clues regarding document identifiers, or consult with the XMPP Extensions Editor team).
The <supersedes/>, <supersededby/>, <shortname/>, and <schemaloc/> elements are for use by the XMPP Extensions Editor team; however, if your document supersedes an existing XMPP Extension Protocol, feel free to include a <spec/> child element specifying the document identifier (e.g., Roster Item Exchange (XEP-0093) [15]) for the protocol that is being superseded.
Include one <author/> element for each co-author. Note well that the <firstname/> and <surname/> elements are REQUIRED per XMPP Extension Protocols (XEP-0001) [16], as is some combination of the <email/>, <jid/>, and <uri/> elements so that appropriate contact information is available.
Include one <revision/> element for each revision of your document. The XML character data of the <version/> element SHOULD be "0.0.1" for your initial submission to the XMPP Extensions Editor team, and the <remark/> SHOULD be "First draft."; for each revision, you will include another <revision/> element (place it *before* the existing <revision/> elements) and iterate the <version/> element (e.g., "0.0.2" after "0.0.1" or "0.10" after "0.9"). The format for the <date/> element is yyyy-mm-dd.

5.3 File Contents

Aside from the metadata in the <header/> element (see above), a XEP file is a series of sections, arranged in a hierarchy (<section1/> is a top-level section, within which you can nest <section2/> sections, and so on down to <section4/>). The title of a section is captured in the 'topic' attribute. You should also include an 'anchor' attribute so that you can link to page fragments from within your document. The allowable elements within a section element probably look familiar from XHTML: <p/> for paragraphs, <ol/> and <ul/> for ordered and unordered lists, and so on.
The <example/> and <code/> elements are used to show protocol snippets; the <example/> element SHOULD possess a 'caption' attribute that describes the example, whereas the <code/> element does not. Define an XML CDATA section within both of these elements so that you do not need to escape the '<' and '>' characters in your sample XML stanzas, since this makes life

[14]XEP-0175: Best Practices for Use of SASL ANONYMOUS <https://xmpp.org/extensions/xep-0175.html>.
[15]XEP-0093: Roster Item Exchange <https://xmpp.org/extensions/xep-0093.html>.
[16]XEP-0001: XMPP Extension Protocols <https://xmpp.org/extensions/xep-0001.html>.

much easier for author and editor alike (see the markup in existing XEP specifications).
The <p/> and <li/> elements can also contain more markup that is familiar from XHTML, such as the <img/> element. Note that hyperlinks are of the form <link url='foo'>bar</link> rather than <a href='foo'>bar</a> (the reasons for this are lost in the mists of time and it is too late to change it now, so you'll just have to adjust). If needed, you can also use inline structural and presentational markup such as <em/>, <strong/>, <tt/>, <cite/>, and <span/> within the <p/> and <li/> elements.
You may also include tables (these are helpful for listing error codes and such). The <table/> element SHOULD possess a 'caption' attribute that describes the table's contents. Standard XHTML table structure applies (<tr/> defines a row, which contains <th/> elements for header rows and <td/> elements for data rows), and the 'colspan' and 'rowspan' attributes are also available if you need them. Table presentation (such as cellpadding and cellspacing) is handled by the XSLT and CSS stylesheets. However keep in mind that tables weren't meant to display a huge amount of text.
The xep.xsl file performs all sorts of magic in converting your XML file into HTML, including creation of the front matter, table of contents, section numbering, notes, and revision history. Feel free to submit patches for this file, but do not commit your modified version to source control.
Although HTML is the primary publishing format for XEPs, since 2009 the XMPP Extensions Editor team has also published XEPs in the form of PDF file. Keeping that in mind, try to avoid tables with too many columns, which might require wider paper than is normal.

6 The Sections of a XEP Document

Most XEP specifications will have most of the following sections, usually in something like the order shown. Other sections may be appropriate (e.g., XHTML-IM (XEP-0071) [17] has a section for W3C Considerations). Use your best judgment regarding the sections you need in order to make your argument, or consult with the XMPP Extensions Editor team regarding your needs.

6.1 Introduction

The introduction to a XEP document is quite important since it provides the rationale for considering the proposal. In particular, the introduction SHOULD include information such as the following:

1. Tasks that users currently cannot complete because we are lacking the protocol you propose. (Note: Users are not just IM users, but any person, system, or application that could gain value from interacting with other entities over XMPP networks.)

[17] XEP-0071: XHTML-IM <https://xmpp.org/extensions/xep-0071.html>.

2. Other projects or protocols and how XMPP technologies could interface with them because of your proposed protocol (e.g., XML-RPC, SOAP).
3. A comparison between XMPP technologies and "the competition" (e.g., other IM systems or messaging protocols) describing holes in the XMPP protocol stack that need to be filled in order to offer similar functionality.
4. The relevant history of thinking within the XMPP community.
5. Real-world examples of problems the protocol can solve.

6.2 Requirements

Every XEP document SHOULD include a section describing the requirements being addressed by the document. This information is critically important, because it clearly defines the scope of the proposal as well as any relevant constraints on the protocol design.

6.3 Glossary

If your XEP document uses terms that may not be familiar to the reader, please define them in a glossary.
The preferred layout for a glossary is a definition list using the HTML <dl> tag (see existing XEPs for examples).

6.4 Use Cases

It is recommended that document authors structure their proposals according to the use cases that the proposal will address. [18] We have found that use cases force authors to focus on functionality rather than "protocol for the sake of protocol". It is also helpful to sort use cases by actor, as is done in Multi-User Chat (XEP-0045) [19], for example. Include one subsection for each use case.
When writing use cases and the associated protocols, make sure to:

- Clearly define the success scenarios, alternate flows, and possible errors.
- Describe the expected behavior of XMPP clients, servers, and components when using this protocol.
- Include lots of protocol examples. [20]

[18] A good introduction to use cases may be found at <http://www.pols.co.uk/usecasezone/>.
[19] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[20] Our mantra is: "We put the example in example.com!"

We repeat: include lots of protocol examples. Examples help not only implementers but also those who will review your proposal in the Standards SIG and XMPP Council. You get extra credit with the XMPP Extensions Editor team if you follow Jabber tradition by using characters and situations from the plays of Shakespeare:

Listing 1: An Example from Shakespeare

```
<message
    from='juliet@capulet.com/balcony'
    to='romeo@montague.net/garden'
    type='chat'>
  <body>Wherefore art thou, Romeo?</body>
</message>
```

6.5 Error Codes

If your proposal defines a number of error and status codes (as is done in Multi-User Chat (XEP-0045) [21]), it is a good idea to include a table of all the codes defined in your document.

6.6 Business Rules

You may want to include a section describing various business rules (essentially, a variety of MUSTs, SHOULDs, and MAYs regarding application behavior). This is not required but can be helpful to implementers.

6.7 Implementation Notes

You may want to include a section devoted to implementation notes. Again, this is not required but can be helpful to implementers.

6.8 Internationalization Considerations

If there are any internationalization or localization issues related to your proposal (e.g., whether to include the 'xml:lang' attribute), define them in this optional section.

6.9 Security Considerations

Your proposal MUST include a section entitled "Security Considerations". Even if there are no security features or concerns related to your proposal, you MUST note that fact. For helpful

[21] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.

guidelines, refer to RFC 3552 [22]; the core XMPP specification (RFC 6120 [23]) also includes a very thorough security considerations section that can be used as an examplar.

6.10 IANA Considerations

This section is REQUIRED. The IANA is the Internet Assigned Numbers Authority (IANA) [24], the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. Most proposals do not require interaction with the IANA, in which case the text of this section SHOULD read ”This document requires no interaction with the Internet Assigned Numbers Authority (IANA).” If your proposal requires interaction with the IANA, discuss the matter with the XMPP Extensions Editor team in their role as the XMPP Registrar. Do not contact the IANA on your own!

6.11 XMPP Registrar Considerations

This section is REQUIRED. The XMPP Registrar [25] maintains a list of reserved XMPP protocol namespaces as well as registries of parameters used in the context of protocols approved by the XMPP Standards Foundation. If your proposal does not require interaction with the XMPP Registrar, the text of this section SHOULD read ”No namespaces or parameters need to be registered with the XMPP Registrar as a result of this document.” Refer to Draft or Final XEPs for appropriate text in other cases, or consult with the XMPP Extensions Editor team in their role as the XMPP Registrar.

6.12 XML Schema

An XML Schema is required in order for protocols to be approved by the XMPP Council. The XMPP Extensions Editor team can assist you in defining an XML Schema for the protocol you are proposing.

6.13 Acknowledgements

Most XEP documents end with a section thanking non-authors who have made significant contributions or who have provided feedback regarding the specification.

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

[23] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

[24] 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/>.

[25] 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/>.

7 XEP Styleguide

XMPP Extension Protocols are written in English. It is not expected that you will be a fine prose writer, but try to write in a clear, easily-understood fashion. The XMPP Extensions Editor team will correct any errors of grammar, spelling [26], punctuation, and usage they may find in your proposal, but might not do so until your proposal is in the XMPP Council's queue for advancement to Draft or Active. In addition, the XMPP Extensions Editor team reserves the right to improve phrases that are unclear or infelicitous, move sections around, modify examples to use Shakespearean characters, and otherwise improve the argument and logical flow of your proposal (naturally, without changing the meaning).
The following styleguide is provided to supplement the standard English styleguides, such as The Elements of Style [27] and The Chicago Manual of Style [28]; please refer to those resources for information about common English (especially American English) usage and to this styleguide for XEP-specific guidelines.

7.1 Attributes

When talking about an attribute by name, refer to it in single quotes. Example: the 'to' attribute.
When talking about the value of an attribute, refer to it in double quotes. Example: the value of the 'subscription' attribute is "both".
Elements *possess* attributes and *contain* character data and/or child elements; do not confuse these terms.

7.2 Code Examples

In examples, use single quotes rather than double quotes; they are more readable.
To show the hierarchy of XML elements, indent two spaces for every level.
If an element possesses multiple attributes, please show them in the order dictated by Canonical XML [29].
If an element possesses a large number of attributes, include a line break before each attribute and indent them so that they are vertically aligned for readability.
If the XML character data of an element is long, include line breaks and indent by two spaces.
Examples are the major source of right-scrolling in our HTML output files. Right-scrolling is evil. Therefore, adjust your example layouts accordingly (line widths should be no more than

[26]With all due respect to authors in other parts of the world, XMPP Extension Protocols follow American spelling conventions; thus "authorisation" will be changed to "authorization" and such.
[27]See <http://en.wikipedia.org/wiki/The_Elements_of_Style>.
[28]See <http://en.wikipedia.org/wiki/The_Chicago_Manual_of_Style>.
[29]Canonical XML 1.0 <http://www.w3.org/TR/xml-c14n>.

110 characters or so).
Example:

```
<iq from='darkcave@macbeth.shakespeare.lit'
    id='config1'
    to='crone1@shakespeare.lit/desktop'
    type='result'>
  <query xmlns='http://jabber.org/protocol/muc#roomconfig'>
    <x xmlns='jabber:x:data' type='form'>
      <title>Configuration for "darkcave" Room</title>
      <instructions>
        Please complete this form to make changes to the configuration
        of your room; to add room owners and administrators, use the
        appropriate room commands rather than this form.
      </instructions>
      <field type='hidden' var='FORM_TYPE'>
        <value>http://jabber.org/protocol/muc#roomconfig</value>
      </field>
  </query>
</iq>
```

Some examples include strings that are the output of a hashing algorithm such as SHA-1 (see RFC 3174 [30]) or SHA-256 (see Use of Cryptographic Hash Functions in XMPP (XEP-0300) [31]). An easy way to generate these is to use the OpenSSL "dgst" command to generate the hash. For example, the following command will generate the SHA-1 hash "9f5f9fdab9da7fc12e3c52b258acbcb4bb8e66ac":

```
echo -n 'bard@shakespeare.lit' | openssl dgst -hex -sha1
```

Some examples (e.g., SASL examples) include strings that are encoded using Base64 (see RFC 4648 [32]). An easy way to generate these is to use the OpenSSL "enc" command to generate the base64-encoded equivalent. For example, the following command will generate the base64-encoded string "YmFyZEBzaGFrZXNwZWFyZS5saXQ=":

```
echo -n 'bard@shakespeare.lit' | openssl enc -nopad -base64
```

7.3 Conformance Terms

Conformance terms (e.g,, "MUST" and "SHOULD") are specified in RFC 2119. Use them. When such terms are not in ALL CAPS, the special conformance sense does not apply (although it is preferable to use terms such as 'might' instead of 'may' and 'ought' instead of 'should').

[30]RFC 3174: US Secure Hash Algorithm 1 (SHA1) <http://tools.ietf.org/html/rfc3174>.
[31]XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>.
[32]RFC 4648: The Base16, Base32, and Base64 Data Encodings <http://tools.ietf.org/html/rfc4648>.

7.4 Elements

When talking about an element by name, refer to it as an empty XML element. Example: the <query/> element.
The top-level <message/>, <presence/>, and <iq/> elements are actually XML stanzas (see RFC 6120 [33]); please refer to them as stanzas, not elements.
Elements *possess* attributes and *contain* character data and/or child elements; do not confuse these terms.
Do not use the term "tag" when you mean "element".
Do not add a possessive to the element itself. Negative example: the <body/>'s character data. Positive example: the <body/> element's character data.
Note: There are shortcuts for stanza names and some common element names in the xep.ent file.

7.5 Errors

When talking about an error condition, use the XML element names defined in RFC 6120 [34] rather than the old HTTP-style code numbers. Example: the <feature-not-implemented/> error.
Note: There are shortcuts for the stanza errors in the xep.ent file.

7.6 Namespaces

When talking about a namespace by name, refer to it in single quotes. Example: the 'jabber:iq:roster' namespace.
An element or attribute is *qualified by* (rather than "scoped by" or "in") a particular namespace.

7.7 Quotes

For precision, the XSF places all punctuation outside the quotation marks unless one is quoting text that includes the punctuation (this is known as "logical punctuation"). Example: the port used for client communications is "5222".

8 Acknowledgements

Thanks to Tobias Markmann and Kevin Smith for their input.

[33] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[34] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.