XEP-0393: Message Styling

Sam Whited
mailto:sam@samwhited.com
xmpp:sam@samwhited.com
https://blog.samwhited.com/

2021-04-04
Version 1.1.1

Status	Type	Short Name
Draft	Standards Track	styling

This specification defines a formatted text syntax for use in instant messages with simple text styling.

Contents

1 Introduction

Historically, XMPP has had no system for simple text styling. Instead, specifications like XHTML-IM (XEP-0071) [1] that require full layout engines have been used, leading to numerous security issues with implementations. Some entities have also performed their own styling based on identifiers in the body. While this has worked well in the past, it is not interoperable and leads to entities each supporting their own informal styling languages.
This specification aims to provide a single, interoperable formatted text syntax that can be used by entities that do not require full layout engines.

2 Requirements

- Clients that do not support this specification MUST still be able to receive messages sent by clients using this specification and display them in a human-readable form.
- Clients that support this specification MUST NOT be required to use a layout engine such as HTML or LaTeX.
- Messages formatted using this specification MUST NOT hinder readability on receiving clients regardless of client background color, contrast, or window size.
- Messages formatted using this specification MUST NOT hinder readability by users with color vision deficiency or impaired vision.
- Messages formatted with this specification MUST render correctly in locales with right-to-left (RTL) layouts without causing confusion.
- Clients that support this specification MUST NOT be required to extract metadata unrelated to formatting or text style from the message.
- Servers MUST NOT need to implement any new functionality for this specification to be supported.

3 Use Cases

- As a user sending an instant message to a friend, I want to be able to emphasize an important part of my message.
- As a software developer, I want to be able to send code as pre-formatted, monospace, block or inline text to another developer.
- As a multi-user chat user I want to add context to my reply by quoting an earlier message in the chat.

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

4 Glossary

Many important terms used in this document are defined in Unicode [2]. The terms ”left-to-right” (LTR) and ”right-to-left” (RTL) are defined in Unicode Standard Annex #9 [3]. The term ”formatted text” is defined in RFC 7764 [4].

Block Any chunk of text that can be parsed unambiguously in one pass. Blocks may contain one or more children which may be other blocks or spans. For example:

A single line of text comprising one or more spans A block quotation A preformatted code block

Formal markup language A structured markup language such as LaTeX, SGML, HTML, or XML that is formally defined and may include metadata unrelated to formatting or text style.

Plain text Text that does not convey any particular formatting or interpretation of the text by computer programs.

Span A group of text that may be rendered inline alongside other spans. Spans may be either plain text with no formatting applied, or may be formatted text that is enclosed by two styling directives. Some spans may contain child spans.

Styling directive A character or set of characters that indicates the beginning of a span or block. For example, in certain contexts the characters ’*’ (U+002A ASTERISK), and ’_-’ (U+005F LOW LINE) may be styling directives that indicate the beginning of a strong or emphasis span and the string ”“‘ (U+0060 GRAVE ACCENT) may be a styling directive that indicate the beginning of a preformatted code block.

Whitespace character Any Unicode scalar value which has the property ”White_Space” or is in category Z in the Unicode Character Database.

5 Discovering Support

Clients that support message styling MUST advertise the fact by including a feature of ”urn::xmpp:styling:0” in response to Service Discovery (XEP-0030) [5] information requests and in their Entity Capabilities (XEP-0115) [6] profiles.

[2]The Unicode Standard, The Unicode Consortium <http://www.unicode.org/versions/latest/>.

[3]Unicode Standard Annex #9, ”Unicode Bidirectional Algorithm”, edited by Mark Davis, Aharon Lanin, and Andrew Glass. An integral part of The Unicode Standard, <http://unicode.org/reports/tr9/>.

[4]RFC 7764: Guidance on Markdown: Design Philosophies, Stability Strategies, and Select Registrations <http://tools.ietf.org/html/rfc7764>.

[5]XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

[6]XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

Listing 1: Disco info response

```
<query xmlns='http://jabber.org/protocol/disco#info'>…

  <feature var='urn:xmpp:styling:0' />…

</query>
```

6 Business Rules

6.1 Blocks

Parsers implementing message styling will first parse blocks and then parse child blocks or spans if allowed by the specific block type.

6.1.1 Plain

Individual lines of text that are not inside of a preformatted text block are considered a ”plain” block. Plain blocks are not bound by styling directives and do not imply formatting themselves, but they may contain spans which imply formatting. Plain blocks may not contain child blocks.

Listing 2: Plain block text

```
<body>
  There are three blocks in this body, one per line,
  but there is no *formatting
  as spans* may not escape blocks.
</body>
```

6.1.2 Preformatted Text

A preformatted text block is started by a line beginning with ”‘‘‘” (U+0060 GRAVE ACCENT), and ended by a line containing only three grave accents or the end of the parent block (whichever comes first). Preformatted text blocks cannot contain child blocks or spans. Text inside a preformatted block SHOULD be displayed in a monospace font.

Listing 3: Preformatted block text

```
<body>
  ‘{}‘{}‘ignored
  (println &quot;Hello, world!&quot;)
  ‘{}‘{}‘

  This should show up as monospace, preformatted text
```

```
</body>
```

Listing 4: No closing preformatted text sequence

```
<body>
  &gt; `{}`{}`
  &gt; (println &quot;Hello, world!&quot;)

  The entire blockquote is a preformatted text block, but this line
  is plaintext!
</body>
```

6.1.3 Quotations

A quotation is indicated by one or more lines with a byte stream beginning with a '>' (U+003E GREATER-THAN SIGN). They are terminated by the first new line that is not followed by a greater-than sign, or the end of the parent block (whichever comes first). Block quotes may contain any child block, including other quotations. Lines inside the block quote MUST have the first leading whitespace character trimmed before parsing the child block. It is RECOMMENDED that text inside of a block quote be indented or distinguished from the surrounding text in some other way.

Listing 5: Quotation (LTR)

```
<body>
  &gt; That that is, is.

  Said the old hermit of Prague.
</body>
```

Listing 6: Nested Quotation

```
<body>
  &gt;&gt; That that is, is.
  &gt; Said the old hermit of Prague.

  Who?
</body>
```

6.2 Spans

Spans are always the children of blocks and may not escape from their containing block. Matches of spans between two styling directives MUST contain some text between the two directives, otherwise neither directive is valid. The opening styling directive MUST be located at the beginning of the parent block, after a whitespace character, or after a different opening styling directive. The opening styling directive MUST NOT be followed by a whitespace character and the closing styling directive MUST NOT be preceeded by a whitespace character. Spans are always parsed from the beginning of the byte stream to the end and are lazily matched. Characters that would be styling directives but do not follow these rules are not considered when matching and thus may be present between two other styling directives. For example, each of the following would be styled as indicated:

- plain span
- ***strong span***
- plain _*emphasis*_ plain
- ‘pre‘ plain ***strong***
- ***strong***plain*
- * plain ***strong***

Nothing would be styled in the following messages (where ”\n” represents a new line):

- not strong*
- *not strong
- *not \n strong*
- *not *strong
- **
- ***
- ****

6.2.1 Plain

Any text inside of a block that is not part of another span is implicitly considered to be inside of a ”plain text” span. In the following example the plain span is everything before the first ”*”.

Listing 7: Plain

```
<body>
  Two spans, both *alike in dignity*
</body>
```

6.2.2 Emphasis

Text enclosed by '_' (U+005F LOW LINE) is emphasized and SHOULD be displayed in italics.

Listing 8: Italic

```
<body>
  The full title is _Twelfth Night, or What You Will_ but
  _most_ people shorten it.
</body>
```

6.2.3 Strong Emphasis

Text enclosed by '*' (U+002A ASTERISK) is strongly emphasized and SHOULD be displayed with a heavier font weight than the surrounding text (bold).

Listing 9: Strong

```
<body>
  The full title is "Twelfth␣Night,␣or␣What␣You␣Will" but
  *most* people shorten it.
</body>
```

6.2.4 Strike through

Text enclosed by '~' (U+007E TILDE) SHOULD be displayed with a horizontal line through the middle.

Listing 10: Strike through

```
<body>
  Everyone ~dis~likes cake.
</body>
```

6.2.5 Preformatted Span

Text enclosed by a "' (U+0060 GRAVE ACCENT) is a preformatted span SHOULD be displayed inline in a monospace font. A preformatted span may only contain a single plain span. Inline formatting directives inside the preformatted span are not rendered. For example, the following all contain valid preformatted spans:

- This is ‘monospace‘
- This is ‘*monospace*‘

- This is ***‘monospace and bold‘***

Listing 11: Monospace text

```
<body>
  Wow, I can write in ‘monospace‘!
</body>
```

7 Disabling Styling

On rare occasions styling hints may conflict with the contents of a message. For example, if the user sends the emoji ”> _ <” it would be interpreted as a block quote. Senders may indicate to the receiver that a particular message SHOULD NOT be styled by adding an empty <unstyled> element qualified by the ”urn:xmpp:styling:0” namespace.

Listing 12: Sender indicates that styling is disabled

```
<message>
  <body>&gt; _ &lt;</body>
  <unstyled xmlns=”urn:xmpp:styling:0”/>
</message>
```

8 Implementation Notes

This document does not define a regular grammar and thus styling cannot be matched by a regular expression. Instead, a simple parser can be constructed by first parsing all text into blocks and then recursively parsing the child-blocks inside block quotations, the spans inside individual lines, and by returning the text inside preformatted blocks without modification.
It is RECOMMENDED that styling directives be displayed and formatted in the same manner as the text they apply to. For example, the string ”*emphasis*” would be rendered as ”***emphasis***”.
This specification does not provide a mechanism for removing styling from individual spans or blocks within a styled message. Implementations are free to implement their own workarounds, for example by inserting Unicode non-printable characters to invalidate styling directives, but no specific technique is known to be widely supported.

9 Accessibility Considerations

When displaying styling directives, developers should ensure sufficient contrast so that visually impaired users are able to distinguish the styling directives from the background color. Care should also be taken to ensure that formatting designed to differentiate styling

directives from surrounding text does not make the text more difficult to read for visually impaired users.
Styled text may be rendered poorly by screen readers. When applying formatting it may be desirable to include directives to exclude styling directives from being read, or to add markers to the final output that have semantic meaning for screen readers. For example, in an web based client an emphasis span might be converted to an HTML <em> element.

10 Security Considerations

Authors of message styling parsers should take care that improperly formatted messages cannot lead to buffer overruns or code execution.
Though message styling is not compatible with Markdown, some of its styles are similar. Markdown parsers often allow arbitrary HTML to be inserted into documents and therefore MUST NOT be used to parse the format defined in this document even if they are flexible enough to do so.

11 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [7].

12 XMPP Registrar Considerations

12.1 Protocol Namespaces

This specification defines the following XML namespace:

- urn:xmpp:styling:0

Upon advancement of this specification from a status of Experimental to a status of Draft, the XMPP Registrar [8] shall add the foregoing namespace to the registries located at <https://xmpp.org/registrar/disco-features.html>, as described in Section 4 of XMPP Registrar Function (XEP-0053) [9].

[7]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/>.
[8]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/>.
[9]XEP-0053: XMPP Registrar Function <https://xmpp.org/extensions/xep-0053.html>.

```
<var>
  <name>urn:xmpp:styling:0</name>
  <desc>Support for rendering message styling.</desc>
  <doc>&xep0393;</doc>
</var>
```

The XMPP Registrar [10] shall also add the foregoing namespace to the Jabber/XMPP Protocol Namespaces Registry located at <https://xmpp.org/registrar/namespaces.html>. Upon advancement of this specification from a status of Experimental to a status of Draft, the XMPP Registrar [11] shall remove the provisional status from this registry entry.

```
<ns>
  <name>urn:xmpp:styling:0</name>
  <doc>&xep0393;</doc>
  <status>provisional</status>
</ns>
```

12.2 Namespace Versioning

If the protocol defined in this specification undergoes a revision that is not fully backwards-compatible with an older version, the XMPP Registrar shall increment the protocol version number found at the end of the XML namespaces defined herein, as described in Section 4 of XEP-0053.

13 XML Schema

This document does not define any new XML structure requiring a schema.

14 Acknowledgements

The author wishes to thank Kevin Smith for his review and feedback.

[10]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/>.

[11]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/>.