XEP-0022: Message Events

Jeremie Miller
mailto:jer@jabber.org
xmpp:jer@jabber.org

DJ Adams
mailto:dj.adams@pobox.com
xmpp:dj@gnu.mine.nu

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

2009-05-27
Version 1.4

Status	Type	Short Name
Obsolete	Historical	x-event

This document defines an XMPP protocol extension used to request and respond to events relating to the delivery, display, and composition of messages. Note: This specification has been obsoleted in favor of XEP-0085 and XEP-0184.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

The 'jabber:x:event' namespace defines extensions used to request and respond to events relating to the delivery, display, and composition of messages.
By attaching a jabber:x:event extension to a <message/> element, the sender can track stages in the delivery of that message to its recipient.
Note: More modern protocol extensions for this functionality have been defined in Chat State Notifications (XEP-0085) [1] for the composing and offline events and in Message Delivery Receipts (XEP-0184) [2] for the delivered and displayed events; those specifications supersede this one.

2 The Events

There are four message events currently defined in this namespace:

2.1 Offline

Indicates that the message has been stored offline by the intended recipient's server. This event is triggered only if the intended recipient's server supports offline storage, has that support enabled, and the recipient is offline when the server receives the message for delivery.

2.2 Delivered

Indicates that the message has been delivered to the recipient. This signifies that the message has reached the recipient's Jabber client, but does not necessarily mean that the message has been displayed. This event is to be raised by the Jabber client.

2.3 Displayed

Once the message has been received by the recipient's Jabber client, it may be displayed to the user. This event indicates that the message has been displayed, and is to be raised by the Jabber client. Even if a message is displayed multiple times, this event should be raised only once.

2.4 Composing

In threaded chat conversations, this indicates that the recipient is composing a reply to a message. The event is to be raised by the recipient's Jabber client. A Jabber client is allowed to

[1] XEP-0085: Chat State Notifications <https://xmpp.org/extensions/xep-0085.html>.
[2] XEP-0184: Message Delivery Receipts <https://xmpp.org/extensions/xep-0184.html>.

raise this event multiple times in response to the same request, providing the original event is cancelled first.

3 Usage

Extensions qualified by the jabber:x:event namespace may be used only in the context of <message/> elements. That is, event information should be requested, and given in response, in relation to <message/> elements only, and not <presence/> or <iq/> elements.
Event information should only be sent in response to a request for that information. Unsolicited event information is illegal. In addition, a client should not request message event information from a correspondent if it is known (for example through the results of a previous browse request) that the correspondent does not support message events.
Any request for the offline event in a message that has been stored offline must be removed by the server before the message is forwarded to the Jabber client. This means that any <offline/> tag should be removed from the extension.

3.1 Requesting Event Notifications

Event notifications are requested by attaching an extension qualified by the jabber:x:event namespace to a <message/> element. A tag representing each event type requested for that message should be placed within the extension. Only one jabber:x:event extension may be attached to a <message/> element, but multiple event types may be requested within that one extension. The tags representing each of the event types are <offline/>, <delivered/>, <displayed/>, and <composing/>.
An example of a <message/> element with a jabber:x:event extension is shown here.

Listing 1: Requesting notification of offline storage and delivery for a message

```
<message to='romeo@montague.net' id='message22'>
  <body>Art thou not Romeo, and a Montague?</body>
  <x xmlns='jabber:x:event'>
    <offline/>
    <delivered/>
    <composing/>
  </x>
</message>
```

Here we see the sender wants to be notified if the message is stored offline (by the server), notified when the message is delivered (to the client), and notified if the recipient begins replying to the message. In this case, the sender will potentially receive three events based on this request. The first if the recipient is offline and the message is stored on the server, the second when the recipient becomes available and the message is delivered, and the third if the recipient begins composing a reply to the message.

Note that the <message/> element requesting event notification contains an 'id' attribute. While these attributes are optional in the Jabber protocol, messages that contain event notification requests MUST contain an 'id' attribute so that raised events may be matched up with their original requests.

3.2 Raising Events

If the message is stored by the server, the server must raise the requested event (offline) by sending a message to the sender as shown in this example:

Listing 2: Raising the offline event

```
<message
    from='romeo@montague.net'
    to='juliet@capulet.com/balcony'>
  <x xmlns='jabber:x:event'>
    <offline/>
    <id>message22</id>
  </x>
</message>
```

When raising an event, the raiser must send a <message/> element constructed according to the following rules:

- The element must contain only a jabber:x:event extension. Standard message elements such as <subject/>, <body/>, MUST NOT be included.
- The extension must contain one tag representing the event being raised. For example, if the offline event is being raised, an <offline/> tag must be included. (The events are temporally exclusive, thus only one event tag should ever be included.)
- The extension must also contain an <id/> tag. The contents of this tag MUST be the value of the 'id' *attribute* of the original message to which this event notification pertains. (If no 'id' attribute was included in the original message, then the <id/> tag must still be included with no CDATA.)
- The message's from attribute should be set to the recipient of the original message for which the event is being raised. (This is an issue more relevant for the server, in responding to the offline event, because clients should rely on the server to stamp the elements that they send out with a from attribute.)
- The link between the original message for which the event is being raised, and the message containing that raised event, is the <id/> tag in the jabber:x:event extension of the message containing that raised event, that points to the id attribute of the original message.

3.3 The Composing Event

The composing event is slightly different from the other events in that it can be raised and cancelled multiple times. This is to allow the reflection of what actually is happening when a user replies to a message; he may start composing a reply, which would trigger the composing event, get halfway through, and stop (by some definition of "stop", which may be implementation-specific). At this stage the event is no longer valid, or at least doesn't make much sense. So the client may send a cancellation for the composing event just raised.
The cancellation is raised by sending another jabber:x:event; however, in contrast to how the events are usually raised, no <composing/> tag is sent, just an <id/> tag, like this:

Listing 3: Romeo begins to compose a reply

```
<message
    from='romeo@montague.net'
    to='juliet@capulet.com/balcony'>
  <x xmlns='jabber:x:event'>
    <composing/>
    <id>message22</id>
  </x>
</message>
```

Listing 4: Romeo pauses to reflect before answering, thus cancelling the composing event

```
<message
    from='romeo@montague.net'
    to='juliet@capulet.com/balcony'>
  <x xmlns='jabber:x:event'>
    <id>message22</id>
  </x>
</message>
```

The lack of an <composing/> tag (and any other event tag) signifies that the composing event, indicated previously by the id value, has been cancelled. In this example, the composing event being cancelled is that event that was previously raised with the id of message22. After cancelling a composing event, a new one may be raised, following the same rules as before, when the typing of the reply is resumed.

4 Examples

This section contains a number of examples to illustrate the full flow of messages, event notifications, and event cancellations for a fictional conversation.

Listing 5: Juliet sends a message to Romeo and requests all event types

```
SEND: <message to='romeo@montague.net' id='message22'>
      <body>Art thou not Romeo, and a Montague?</body>
      <x xmlns='jabber:x:event'>
        <offline/>
        <delivered/>
        <displayed/>
        <composing/>
      </x>
    </message>
```

Romeo temporarily loses his wireless connection in the Capulet's orchard and therefore his message is stored offline by the montague.net server, which generates the offline event and sends that notification to Juliet.

Listing 6: Receiving the offline event

```
RECV: <message
          from='romeo@montague.net'
          to='juliet@capulet.com/balcony'>
      <x xmlns='jabber:x:event'>
        <offline/>
        <id>message22</id>
      </x>
    </message>
```

Romeo reconnects and the message is delivered to his Jabber client, which generates a delivered event and sends it to Juliet's client.

Listing 7: Juliet receives notification of message delivery

```
RECV: <message
          from='romeo@montague.net/orchard'
          to='juliet@capulet.com/balcony'>
      <x xmlns='jabber:x:event'>
        <delivered/>
        <id>message22</id>
      </x>
    </message>
```

Romeo's Jabber client displays the message and sends a displayed event to Juliet's client.

Listing 8: Juliet receives notification of message display

```
RECV: <message
          from='romeo@montague.net/orchard'
          to='juliet@capulet.com/balcony'>
      <x xmlns='jabber:x:event'>
        <displayed/>
        <id>message22</id>
```

```
      </x>
    </message>
```

Romeo begins composing a reply to Juliet's heartfelt question, and his Jabber client notifies Juliet that he is composing a reply.

Listing 9: Juliet receives notification of message composing

```
RECV: <message
          from='romeo@montague.net/orchard'
          to='juliet@capulet.com/balcony'>
        <x xmlns='jabber:x:event'>
          <composing/>
          <id>message22</id>
        </x>
      </message>
```

Romeo realizes his reply is too rash and pauses to choose the right words; his Jabber client senses the delay and cancels the previous composing event.

Listing 10: Juliet receives cancellation of message composing

```
RECV: <message
          from='romeo@montague.net/orchard'
          to='juliet@capulet.com/balcony'>
        <x xmlns='jabber:x:event'>
          <id>message22</id>
        </x>
      </message>
```

Romeo starts composing again, and his Jabber client sends a notification to Juliet's client.

Listing 11: Juliet receives a second notification of message composing

```
RECV: <message
          from='romeo@montague.net/orchard'
          to='juliet@capulet.com/balcony'>
        <x xmlns='jabber:x:event'>
          <composing/>
          <id>message22</id>
        </x>
      </message>
```

Romeo finally sends his reply, and requests composing events related to it.

Listing 12: Romeo replies

```
SEND: <message to='juliet@capulet.com' id='GabberMessage43'>
```

```
    <body>Neither, fair saint, if either thee dislike.</body>
    <x xmlns='jabber:x:event'>
      <composing/>
    </x>
  </message>
```

5 Implementation Notes

Compliant implementations SHOULD observe the following business rules:

1. Every outgoing message sent from a compliant client should contain a request for event notifications (if such notifications are desired). The request for notifications cannot be sent just once in a conversation, since it applies to every message sent.
2. When a client receives a request for events from another entity, it should cache the most recent ID.
3. When a user begins replying to a message received from a contact, the user's client should check to see whether events have been requested by the contact for that message and set the CDATA of the <id/> element to the cached ID value.
4. The CDATA of the <id/> element MUST be the same as the value of the 'id' attribute of the notification request.

6 Security Considerations

There are no security features or concerns related to this proposal.

7 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [3].

[3] 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 XMPP Registrar Considerations

No action on the part of the XMPP Registrar [4] is necessary as a result of this document, since 'jabber:x:event' is already a registered protocol namespace.

9 XML Schema

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='jabber:x:event'
    xmlns='jabber:x:event'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0022: http://www.xmpp.org/extensions/xep-0022.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='x'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='offline' minOccurs='0' type='empty'/>
        <xs:element name='delivered' minOccurs='0' type='empty'/>
        <xs:element name='displayed' minOccurs='0' type='empty'/>
        <xs:element name='composing' minOccurs='0' type='empty'/>
        <xs:element name='id' minOccurs='0' type='xs:NMTOKEN'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

[4] 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/>.

10 Open Issues

1. In a Standards Track specification addressing event functionality, it would be desirable to have more cancellation methods for composing events than those defined in this Informational document. For instance, is someone still composing if they become unavailable? This example points to the fact that cancellation of a composing event can either be explicit (the default or desired scenario) or implicit (e.g., through a change in the availability state of a client or the existence of the session associated with the message composition).