



XMPP

XEP-0023: Message Expiration

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

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

Harold Gottschalk
<mailto:heg@imissary.com>
<xmpp:heg@imissary.com>

2009-06-03
Version 1.3

Status	Type	Short Name
Obsolete	Historical	x-expire

This specification documents an historical protocol that was used to specify expiration dates for messages; this protocol has been deprecated in favor of XEP-0079: Advanced Message Processing.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2017 by the [XMPP Standards Foundation](#) (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the "Specification"), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

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	1
2	Specifying a TTL	1
3	Handling XML Stanzas with a TTL	2
3.1	Usage in client space	2
4	Security Considerations	3
5	IANA Considerations	3
6	XMPP Registrar Considerations	3
7	XML Schema	3
8	Open Issues	4

1 Introduction

Note Well: The protocol described herein has been deprecated by the [XMPP Standards Foundation \(XSF\)](#)¹. The recommended protocol for implementing message expiration functionality is now [Advanced Message Processing \(XEP-0079\)](#)².

It is sometimes helpful to indicate that a piece of information has a finite useful life or time-to-live (TTL). In the context of instant messaging, the main use of a TTL is to indicate that a message must or should be used by or read by a certain time, usually because the message has meaning or purpose only within a finite amount of time. In normal usage, such a message should be discarded after the specified time has passed if it has not been used or read by that time.

In Jabber, TTL functionality has been implemented informally using the `jabber:x:expire` namespace. Support for this namespace was added to the [jabberd](#)³ server as well as some clients and components in early 2001. Specifically, that support has involved the following two areas of responsibility:

- The sender of the message is responsible for attaching a `jabber:x:expire` extension to the XML stanza (usually a message).
- Mechanisms involved in the store-and-forward of such a stanza⁴ en route to its intended recipient are responsible for checking the remaining time to live and expiring (discarding) the XML stanza if necessary.

2 Specifying a TTL

An Endpoint can specify a TTL for an XML stanza that it wishes to send by attaching an `<x/>` extension qualified by the `jabber:x:expire` namespace. The extension contains no children, only a 'seconds' attribute that contains a value representing the stanza's TTL, in seconds.

Listing 1: Specifying a 30-minute TTL for a message

```
SEND: <message to='sabine@gnu.mine.nu' id='msg811'>
      <subject>Eccles cakes!</subject>
      <body>
        I've got some freshly made Eccles cakes here, come
        round for one before they all disappear!
```

¹The 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>.

²XEP-0079: Advanced Message Processing <https://xmpp.org/extensions/xep-0079.html>.

³The 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/>.

⁴The best-known example of a mechanism that handles storing and forwarding of XML stanzas is the Jabber Session Manager (JSM) within current Jabber server implementations, specifically the `mod_offline` module. However, expiration of an XML stanza could also be handled by a Jabber client.

```

.....</body>
.....<x_xmlns='jabber:x:expire' _seconds='1800' />
.....</message>

```

3 Handling XML Stanzas with a TTL

Any mechanism that is involved in the storage, forwarding, and general handling of XML stanzas must check for the presence of such an extension and act accordingly, expiring (discarding) any stanzas that have exceeded their TTL lifetime. The `jabber:x:expire` namespace allows for a further attribute inside the `<x/>` extension: `'stored'`. Here, the mechanism can record a value representing when the stanza was committed to storage, so that when the stanza is eventually retrieved for forwarding to the intended recipient, the elapsed time of storage can be calculated. This is to prevent the stanza from being held in 'suspended animation'.

Here we see what the original message looks like after the stanza has been committed to storage and the time of storage recorded:

Listing 2: Recording a storage-time in the extension

```

SEND: <message to='sabine@gnu.mine.nu' id='msg811'>
      <subject>Eccles cakes!</subject>
      <body>
          I've_got_some_freshly_made_Eccles_cakes_here,_come
....._round_for_one_before_they_all_disappear!
.....</body>
.....<x_xmlns='jabber:x:expire'
....._seconds='1800'
....._stored='912830221' />
.....</message>

```

When Sabine attempts to retrieve her offline messages, the store-and-forward mechanism (e.g., `mod_offline`) compares the current time against the stored attribute. If the 1800 seconds have passed, the mechanism should simply drop the message, without notifying either the sender or the intended recipient. No Eccles cakes for Sabine!

3.1 Usage in client space

Although current usage of `jabber:x:expire` is most commonly seen in server implementations to address any TTL requirements of stored messages, Jabber clients can also be seen as handlers of messages that may contain expiration extension information. If a message is received by a Jabber client, and not immediately displayed to the user, the client must check for TTL information and expire the message (rather than display it to the user) if appropriate.

4 Security Considerations

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

5 IANA Considerations

This document requires no interaction with the [Internet Assigned Numbers Authority \(IANA\)](#)⁵.

6 XMPP Registrar Considerations

No action on the part of the [XMPP Registrar](#)⁶ is necessary as a result of this document, since 'jabber:x:expire' is already a registered protocol namespace.

7 XML Schema

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema
  xmlns:xs='http://www.w3.org/2001/XMLSchema'
  targetNamespace='jabber:x:expire'
  xmlns='jabber:x:expire'
  elementFormDefault='qualified'>
  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0023: http://www.xmpp.org/extensions/xep-0023.html
    </xs:documentation>
  </xs:annotation>
  <xs:element name='x'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='empty'>
```

⁵The 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/>.

⁶The 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/>.

```
        <xs:attribute name='seconds' type='xs:unsignedLong' use='
            required' />
        <xs:attribute name='stored' type='xs:unsignedLong' use='
            optional' />
    </xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>

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

8 Open Issues

1. The jabber:x:expire namespace is processed only for delayed messages and only by servers and subsystems which support this informational draft. Therefore it is possible or even likely that a TTL will not be properly handled from the user perspective.
2. A physical, time-based TTL is not implemented by this document, and would not be possible across systems without synchronized time.