XEP-0410: MUC Self-Ping (Schrödinger's Chat)

Georg Lukas
mailto:georg@op-co.de
xmpp:georg@yax.im

2019-09-25
Version 1.1.0

Status	Type	Short Name
Draft	Standards Track	muc-selfping

This protocol extension for XEP-0045 Multi User Chat allows clients to check whether they are still joined to a chatroom.

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 Multi-User Chat (XEP-0045) [1] protocol was not designed to handle s2s interruptions or message loss well. Rather often, the restart of a server or a component causes a client to believe that it is still joined to a given chatroom, while the chatroom service does not know of this occupant.
Existing approaches for re-synchronization are either inefficient (presence updates and "silent" messages are reflected to all occupants, totalling to $O(N^2)$ stanzas per time unit), or mask message / presence losses (the implicit join performed via the deprecated GC1.0 protocol).
This specification aims to provide the most efficient, albeit not the most elegant, way for clients to periodically check whether they are still joined to a chatroom. However, it can not ensure that a client remains joined to a room without any interruptions.

2 Requirements

This specification only makes sense in the context of Multi-User Chat (XEP-0045) [2] chatrooms. It makes use of XMPP Ping (XEP-0199) [3] to perform periodic self-pings.
Server support for this extension is optional, but will significantly improve the reliability with Multi-Session Nicks and mobile clients.

3 Client Self-Presence Check

A typical connection between a client and a Multi-User-Chatroom (MUC) goes through the client-to-server link, possibly a server-to-server link and a typically local server-to-component link. If one of the involved servers or the MUC component is restarted, or one of the links is disturbed for some time, this can lead to the removal of some or all occupants from the affected MUCs, without the clients being informed.
To an occupant, this looks like the MUC is silent (there is no chat activity and no presence changes), making it hard to realize that the connection was interrupted.
To prevent the bad usability effect (message loss, lack of reaction from people in a chatroom), a client needs to actively check whether it is still joined to a MUC.

3.1 Possible Protocol Approaches

There are multiple alternative approaches for a client to test whether it is still joined to a MUC:

[1] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[2] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[3] XEP-0199: XMPP Ping <https://xmpp.org/extensions/xep-0199.html>.

1. **Silent message** (e.g. Chat State Notifications (XEP-0085) [4]): this message will be reflected to all MUC occupants, causing unwanted traffic and potentially waking up mobile devices without reason. If implemented by all clients, this will result in $O(N^2)$ messages to the MUC.

2. **Presence update**: if the MUC service implements the legacy GC1.0 protocol, this will be treated as a join attempt, and the MUC will return the full list of occupants and full room history. The user's client will however miss partial history (other occupants leaving, potentially also messages), and this has the same drawbacks as the first solution.

3. **Private message to self**: the client can send a MUC private message to itself. However, not all MUCs support / allow private messages, and there is no way to differentiate that from the error responses.

4. **Private IQ to self**: the client can send an IQ to its own occupant JID. MUCs typically do not forbid those, and reflect the IQ request to the client (or another client of the same user). Once that client responds to the reflected IQ, the response is delivered to the initiating client as a sign of still being joined.

5. **Dedicated MUC IQ**: a new type of IQ can be deployed to let the client explicitly check whether it is still joined to a MUC. However, this needs to be supported by the server, and the client needs to implement a fallback solution.

The private IQ is the most robust and traffic-efficient solution, and it does not rely on server support. The XMPP Ping (XEP-0199) [5] protocol is appropriate to use for this use case.

3.2 Performing a Self-Ping

After an adequate amount of silence from a given MUC (e.g. 15 minutes), or from all MUCs from a given service domain, a client should initiate a self-ping. If Juliet is joined as JuliC in the characters@chat.shakespeare.lit MUC, her client will send the following ping IQ:

Listing 1: Self-Ping by Juliet's Client

```
<iq from='juliet@capulet.lit/client' id='s2c1' type='get'
      to='characters@chat.shakespeare.lit/JuliC'>
  <ping xmlns='urn:xmpp:ping'/>
</iq>
```

If Juliet's client is not joined, the MUC service will respond with a <not-acceptable> error. Thus, her client can automatically attempt a rejoin.

[4] XEP-0085: Chat State Notifications <https://xmpp.org/extensions/xep-0085.html>.
[5] XEP-0199: XMPP Ping <https://xmpp.org/extensions/xep-0199.html>.

Listing 2: Server Response to a Non-Occupant

```
<iq from='characters@chat.shakespeare.lit/JuliC' id='s2c1' type='error
    '
      to='juliet@capulet.lit/client' >
  <error type="cancel" by="characters@chat.shakespeare.lit">
    <not-acceptable xmlns="urn:ietf:params:xml:ns:xmpp-stanzas" />
  </error>
</iq>
```

If her client is joined, the IQ request will be forwarded to any one of Juliet's joined clients.

Listing 3: Server Reflection of Ping

```
<iq from='characters@chat.shakespeare.lit/JuliC' id='c0ffee-s2c1' type
   ='get'
      to='juliet@capulet.lit/somerandomclient' >
  <ping xmlns='urn:xmpp:ping'/>
</iq>
```

Depending on the other client implementation and its connection status, the IQ will be responded to eventually, in one of these ways, as delivered to the "client" resource:

- **Successful IQ response**: the client is still joined.
- **Error (<service-unavailable> or <feature-not-implemented>)**: the client is joined, but the pinged client does not implement XMPP Ping (XEP-0199) [6].
- **Error (<item-not-found>)**: the client is joined, but the occupant just changed their name (e.g. initiated by a different client).
- **Error (<remote-server-not-found> or <remote-server-timeout>)**: the remote server is unreachable for unspecified reasons; this can be a temporary network failure or a server outage. No decision can be made based on this; Treat like a timeout (see below).
- **Any other error**[7]: the client is probably not joined any more. It should perform a re-join.
- **Timeout (no response)**: the MUC service (or another client) is unreachable. The client may indicate the status to the user and re-attempt the self-ping after some timeout, until it receives either an error or a success response.

[6] XEP-0199: XMPP Ping <https://xmpp.org/extensions/xep-0199.html>.

[7] Different service implementations will send different responses to a client that's not joined. The recommended error code is <not-acceptable>, however some servers will respond with <not-allowed> or <bad-request> as well.

3.3 Server Optimization

The normal routing rules of the self-ping impose two round-trips: first the initial ping from the client to the MUC, then the reflection of the ping and its response (possibly to another client), and finally the response to the initial IQ. If the other client is experiencing network connectivity issues, which is often the case with mobile devices, the ping request might never be responded to.
Therefore, a MUC service supporting this protocol may directly respond to a occupant's Ping request to the occupant's own nickname, as opposed to routing it to any of the occupant's clients. A service implementing this optimization needs to advertise the http://jabber.org/protocol/muc#self-ping-optimization feature in the Service Discovery (XEP-0030) [8] disco#info response on the individual MUC room JIDs, and it MUST respond to a self-ping request as follows:

- **Successful IQ response**: the client is joined to the MUC.
- **Error (<not-acceptable>)**: the client is not joined to the MUC.

Listing 4: MUC Service Advertises Self-Ping Optimization

```
<iq from='darkcave@chat.shakespeare.lit'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <!-{}- ... -{}->
    <feature var='http://jabber.org/protocol/muc#self-ping-
       optimization'/>
  </query>
</iq>
```

4 Implementation Notes

In Multi-Session-Nick scenarios, where multiple clients of the same user are joined as the same occupant, it is possible that another client initiates a nickname change while a ping request is pending. In that case, the ping might be responded to with <item-not-found>.
A client should not perform a self-ping after initiating a nickname change, and before receiving the response to the nickname change from the service, as it is not yet clear whether the new nickname will be accepted.
If a client session is in hibernation (Stream Management (XEP-0198) [9]), the client should defer sending of self-ping requests until it is reconnected and re-authenticated.

[8] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[9] XEP-0198: Stream Management <https://xmpp.org/extensions/xep-0198.html>.

5 Security Considerations

A MUC service implementation should not allow a non-occupant to obtain information about occupants. This is however true irregardless of implementing this specification.

6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [10].

7 XMPP Registrar Considerations

Include "http://jabber.org/protocol/muc#self-ping-optimization" as a valid feature in the Registry of Features.

```
<var>
  <name>http://jabber.org/protocol/muc#self-ping-optimization</name>
  <desc>Support for the MUC self-ping optimization</desc>
  <doc>XEP-0410</doc>
</var>
```

8 XML Schema

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

[10] 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/>.