XEP-0316: MUC Eventing Protocol

Peter Saint-Andre
mailto:xsf@stpeter.im
xmpp:peter@jabber.org
http://stpeter.im/

2013-01-03
Version 0.1

Status	Type	Short Name
Deferred	Standards Track	mep

This specification defines semantics for using the XMPP publish-subscribe protocol to broadcast state change events associated with a Multi-User Chat (MUC) room. This profile of pubsub therefore enables a chatroom to function as a virtual pubsub service, easing the discovery of syndicated data and event notifications associated with such a room.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Just as Personal Eventing Protocol (XEP-0163) [1] defines a profile of Publish-Subscribe (XEP-0060) [2] that enables an instant messaging user to send updates or "events" to other users, this specification defines a profile that enables a room occupant or the chatroom itself to send notifications in the context of Multi-User Chat (XEP-0045) [3] chatroom.
Note: Any use cases, error flows, and other protocols details not described herein are described in XEP-0060. This document merely defines a "subset" or "profile" of XMPP publish-subscribe.

2 Room Notifications

Using the chatroom example from XEP-0045, imagine that the room itself wants to notify the occupants of the coven@chat.shakespeare.lit chatroom about events of interest to the group (say, notifications about multimedia aspects of the multi-user session, such as described in Delivering Conference Information to Jingle Participants (Coin) (XEP-0298) [4]).
When new information is available about the multimedia session (say, calling into a conference "bridge" or starting a Jingle (XEP-0166) [5] session to add a video feed), a multimedia engine might capture that event and inform the chatroom by means of a backend API. The chatroom itself then generates an event notification. That is, a chatroom does not publish events, instead it simply generates them based on data of interest. As a result, everyone in the room who is interested in that kind of data will receive a notification about the event:

Listing 1: Interested occupants receive event notification

```
<message from='coven@chat.shakespeare.lit'
         to='crone1@shakespeare.lit/desktop'
         type='headline'
         id='zns61f38'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='urn:ietf:params:xml:ns:conference-info'>
      <item id='ehs51f40'>
        <conference-info
            xmlns='urn:ietf:params:xml:ns:conference-info'>
          <conference-description>
            <available-media label='4'>
              <type>video</type>
              <status>sendonly</status>
            </available-media>
          </conference-description>
```

[1] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.
[2] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[3] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[4] XEP-0298: Delivering Conference Information to Jingle Participants (Coin) <https://xmpp.org/extensions/xep-0298.html>.
[5] XEP-0166: Jingle <https://xmpp.org/extensions/xep-0166.html>.

```
        </conference-info>
      </item>
    </items>
  </event>
</message>

<message from='coven@chat.shakespeare.lit'
         to='wiccarocks@shakespeare.lit/laptop'
         type='headline'
         id='uh4gs519'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='urn:ietf:params:xml:ns:conference-info'>
      <item id='ehs51f40'>
        <conference-info
            xmlns='urn:ietf:params:xml:ns:conference-info'>
          <conference-description>
            <available-media label='4'>
              <type>video</type>
              <status>sendonly</status>
            </available-media>
          </conference-description>
        </conference-info>
      </item>
    </items>
  </event>
</message>
```

But how do the occupants tell the room that they are interested in knowing what about conference-info events? Whereas generic pubsub services require an explicit subscription to a conference-info node, MEP services support the "filtered-notification" feature from XEP-0060 and obviously share presence (since MUC is based on directed presence in the room) so the "auto-subscribe" feature also applies.

Listing 2: Occupant sends directed presence to join the room, with caps

```
<presence from='crone1.shakespeare.lit/desktop'
          to='coven@chat.shakespeare.lit/firstwitch'>
  <x xmlns='http://jabber.org/protocol/muc'/>
  <c xmlns='http://jabber.org/protocol/caps'
     hash='sha-1'
     node='http://jisti.org'
     ver='054H4A7280JuT6+IroVYxgCAjZo='/>
</presence>
```

That chatroom knows to send conference-info notifications to crone1@shakespeare.lig because when the room unpacks the value of the 'ver' attribute ("054H4A7280JuT6+IroVYxgCAjZo=") in accordance with XEP-0115, it discovers that her client advertises a service discovery feature of "urn:ietf:params:xml:ns:conference-info+notify",

where the "+notify" suffix indicates interest in receiving notifications related to the protocol that precedes the suffix. The server can verify this support if needed by sending a service discovery request to crone1's full JID (see XEP-0115 for details).

3 Occupant Notifications

The foregoing section described how the room itself can inform the occupants about data of interest. However, in MEP any particular occupant can also publish information. An occupant does so by sending a publish-subscribe publish request to the occupant's Occupant JID <room@service/nick> (similar to the way in which publishing via PEP happens by sending a request to the user's bare JID <user@host>). For instance, the following example shows how a room occupant would inform the other occupants about an event of interest.

Listing 3: Occupant Publishes Activity

```
<iq from='wiccarocks@shakespeare.lit/laptop'
    id='vlk3h264'
    to='coven@chat.shakespeare.lit/secondwitch'
    type='set'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='http://jabber.org/protocol/activity'>
      <item>
        <activity xmlns='http://jabber.org/protocol/activity'>
          <other/>
          <text xml:lang='en'>I'll␣give␣thee␣a␣wind.</text>
␣␣␣␣␣␣␣␣</activity>
␣␣␣␣␣␣</item>
␣␣␣␣</publish>
␣␣</pubsub>
</iq>
```

Note: Publishing to an occupant's MEP node happens by sending an explicit publish request to the Occupant JID. Publishing to a user's PEP node MUST NOT trigger a MEP publish request, because PEP and MEP are separate pubsub contexts.
As a result, everyone in the room who is interested in that kind of data will receive a notification about the event (note that even the publisher receives the event, if they have advertised interest in the payload type):

Listing 4: Interested occupants receive event notification

```
<message from='coven@chat.shakespeare.lit/secondwitch'
         to='crone1@shakespeare.lit/desktop'
         type='headline'
         id='hs7bx143'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
```

```
    <items node='http://jabber.org/protocol/activity'>
      <item id='bxg1c27r'>
        <activity xmlns='http://jabber.org/protocol/activity'>
          <other/>
          <text xml:lang='en'>I'll␣give␣thee␣a␣wind.</text>
␣␣␣␣␣␣␣␣</activity>
␣␣␣␣␣␣</item>
␣␣␣␣</items>
␣␣</event>
</message>

<message␣from='coven@chat.shakespeare.lit'
␣␣␣␣␣␣␣␣␣␣to='wiccarocks@shakespeare.lit/laptop'
␣␣␣␣␣␣␣␣␣␣type='headline'
␣␣␣␣␣␣␣␣␣␣id='ke2v1c95'>
␣␣<event␣xmlns='http://jabber.org/protocol/pubsub#event'>
␣␣␣␣<items␣node='http://jabber.org/protocol/activity'>
␣␣␣␣␣␣<item␣id='bxg1c27r'>
␣␣␣␣␣␣␣␣<activity␣xmlns='http://jabber.org/protocol/activity'>
␣␣␣␣␣␣␣␣␣␣<other/>
␣␣␣␣␣␣␣␣␣␣<text␣xml:lang='en'>I'll give thee a wind.</text>
        </activity>
      </item>
    </items>
  </event>
</message>
```

4 Concepts and Approach

MUC eventing via pubsub ("MEP") is based on the following principles:

1. Every room JID and occupant JID a pubsub service.
2. One publisher per node (the chatroom or occupant itself).
3. Use presence (implicit in multi-user chat).
4. Filter notifications based on expressed interest.
5. Smart defaults.

These principles are described more fully below.

4.1 Every Room JID and Occupant JID a Pubsub Service

Treating every MEP-enabled chatroom as a pubsub service simplifies the task of discovering and subscribing to information of interest in or about the room.

4.2 One Publisher Per Node

There is no need for multiple publishers to a MEP service, since by definition only the chatroom itself or the occupant itself publishes information.

4.3 Use Presence

By definition, a chatroom has presence information about the occupants, because they use directed presence to join the room.

4.4 Filtered Notifications

By default, the use of directed presence is used to establish a MEP subscription to the chatroom's eventing data. In order to filter which notifications are sent by the MEP service, the contact's client includes extended Entity Capabilities (XEP-0115) [6] information in the directed presence notifications it sends to the chatroom. Because the MEP-enabled room supports the "filtered-notifications" feature, it sends only those notifications that match the occupant's expressed notification preferences.

4.5 Smart Defaults

Most pubsub configuration options and metadata are not needed for MUC eventing. Instead, MEP services offer smart defaults to simplify node creation and management.

5 Recommended Defaults

A MEP service MUST:

- Support the node discovery, node creation, node deletion, publish item, subscribe, unsubscribe, and item retrieval use cases specified in XEP-0060.
- Support the "auto-create", "auto-subscribe", and "filtered-notifications" features.

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

- Support the ”owner” and ”subscriber” affiliations.
- Support the ”presence” access model and set it to the default.
- Treat the room JID and each occupant JID as a collection node (i.e., as the root collection node for the virtual pubsub service associated with the chatroom or with each occupant JID).
- Default the 'deliver_notifications' configuration option to true (i.e., deliver payloads by default).
- Default the 'send_last_published_item' configuration option to on_sub_and_presence (i.e., send the last published item on subscription and on receipt of presence), treating the receipt of directed presence as equivalent to a subscription.

A PEP service MAY support other use cases, affiliations, access models, and features, but such support is OPTIONAL.

6 Determining Support

If a chatroom supports MEP, it MUST advertise that fact in its responses to Service Discovery (XEP-0030) [7] information (”disco#info”) requests by returning an identity of ”pubsub/mep” and the relevant pubsub features:

Listing 5: A disco#info query

```
<iq type='get'
    from='hag66@shakespeare.lit/pda'
    to='coven@chat.shakespeare.lit'
    id='k3hs5174'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 6: A disco#info response

```
<iq type='result'
    from='coven@chat.shakespeare.lit'
    to='hag66@shakespeare.lit/pda'
    id='k3hs5174'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='conference' type='text'/>
    <identity category='pubsub' type='mep'/>
    ...
    <feature var='http://jabber.org/protocol/disco#info'/>
    <feature var='http://jabber.org/protocol/disco#items'/>
```

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

```
    <feature var='http://jabber.org/protocol/muc'/>
    <feature var='http://jabber.org/protocol/pubsub#access-presence'/>
    <feature var='http://jabber.org/protocol/pubsub#auto-create'/>
    <feature var='http://jabber.org/protocol/pubsub#auto-subscribe'/>
    <feature var='http://jabber.org/protocol/pubsub#config-node'/>
    <feature var='http://jabber.org/protocol/pubsub#create-and-
       configure'/>
    <feature var='http://jabber.org/protocol/pubsub#create-nodes'/>
    <feature var='http://jabber.org/protocol/pubsub#filtered-
       notifications'/>
    <feature var='http://jabber.org/protocol/pubsub#persistent-items'/
       >
    <feature var='http://jabber.org/protocol/pubsub#publish'/>
    <feature var='http://jabber.org/protocol/pubsub#retrieve-items'/>
    <feature var='http://jabber.org/protocol/pubsub#subscribe'/>
    <feature var='muc_passwordprotected'/>
    <feature var='muc_hidden'/>
    <feature var='muc_temporary'/>
    <feature var='muc_open'/>
    <feature var='muc_unmoderated'/>
    <feature var='muc_nonanonymous'/>
    ...
  </query>
</iq>
```

7 Security Considerations

The security considerations of XEP-0045 and XEP-0163 apply.

8 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [8].

[8] 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/>.

9 XMPP Registrar Considerations

9.1 Service Discovery Category/Type

The XMPP Registrar [9] includes a category of "pubsub" in its registry of Service Discovery identities (see <https://xmpp.org/registrar/disco-categories.html>); as a result of this document, the Registrar includes a type of "pep" to that category.
The registry submission is as follows:

```
<category>
  <name>pubsub</name>
  <type>
    <name>mep</name>
    <desc>
      A MUC eventing service that supports the
      publish-subscribe subset defined herein.
    </desc>
    <doc>XEP-0316</doc>
  </type>
</category>
```

10 XML Schema

Because MEP simply reuses the protocol specified in XEP-0060, a separate schema is not needed.

11 Acknowledgements

Thanks to Joe Hildebrand, Matt Miller, and Matthew Wild for their input.

[9] 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/>.