XEP-0317: Hats

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	NOT_YET_ASSIGNED

This specification defines a more extensible model for roles and affiliations in Multi-User Chat rooms.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Multi-User Chat (XEP-0045) [1] defines a widely-implemented XMPP extension for chatrooms, including basic roles and affiliations such as owner, administrator, and moderator. However, in many scenarios it is desirable to define different roles that are appropriate for the relevant application. Examples might include a ”presenter” or a ”scribe” in an online meeting system, a ”representative” or a ”manager” in a customer service application, a ”comms officer” in a military chat system, an ”incident manager” in a first responder system, a ”teacher” or a ”teacher’s assistant” in an online classroom, specialized roles in online games, etc. To prevent confusion with standard MUC roles, these extended roles are call ”hats”, since a participant can ”wear many hats” in a room.

2 Discovery

A MUC service that supports hats MUST advertise a Service Discovery (XEP-0030) [2] feature of ”urn:xmpp:hats:0”.

3 Protocol

3.1 Including a Hat in Presence

MUC already includes a way for the room to signal the roles and affiliations of room occupants. Hats are signalled in a similar way. For example, the following presence notification would be sent by the room for an occupant who is a MUC room moderator but who also has a hat of ”teacher’s assistant” in an online classroom.

Listing 1: Presence With Hat

```
<presence
    from='physicsforpoets@courses.example.edu/Terry'
    id='DE5C66DE-EC7D-4ECB-844A-B717A67CCE3D'
    to='steve@example.edu/tablet'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='owner' role='moderator'/>
  </x>
  <hats xmlns='urn:xmpp:hats:0'>
    <TeacherAssistant xmlns='http://tech.example.edu/hats' displayname
       ='Teacher&amp;apos;␣Assistant' xml:lang='en-us'/>
  </hats>
</presence>
```

[1] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[2] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

Note: The format is open for debate. Possibilities include:

1. XML element in the hats namespace with name as a URI:
   <hat xmlns='urn:xmpp:hats:0' name='http://tech.example.edu/hats#TeacherAssistant' display='Teacher's Assistant' xml:lang='en-us'/>
   Pro: Clients that don't understand the 'http://tech.example.edu/hats#TeacherAssistant' semantics can at least display a human-readable name. Names can be registered with the XMPP Registrar. Also appropriate as a field name in Ad-Hoc Commands.
   Con: Not a very Jabberish way of structuring information.

2. XML element in the hats namespace with name scoped using Clark Notation:
   <hat xmlns='urn:xmpp:hats:0' name='pathhttp://tech.example.edu/hats}TeacherAssistant' display='Teacher's Assistant' xml:lang='en-us'/>
   Pro: Clients that don't understand the 'http://tech.example.edu/hats#TeacherAssistant' semantics can at least display a human-readable name. Names can be registered with the XMPP Registrar. Also integrates well with Ad-Hoc Commands.
   Con: Not a very Jabberish way of structuring information.

3. XML element qualified by custom namespace:
   <TeacherAssistant xmlns='http://tech.example.edu/hats' displayname='Teacher's Assistant' xml:lang='en-us'/>
   Pro: A more Jabberish way to structure information.
   Con: Clients won't show anything if they don't understand the custom namespace.

As noted, a participant can wear many hats. The following example shows a participant who is a MUC room owner and both a "host" and a "presenter" in an online meeting system.

Listing 2: Presence With Multiple Hats

```
<presence
    from='meeting123@meetings.example.com/Harry'
    id='D568A74F-E062-407C-83E9-531E91526516'
    to='someone@example.com/foo'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='owner' role='moderator'/>
  </x>
  <hats xmlns='urn:xmpp:hats:0'>
    <hat displayName='Host' name='http://schemas.example.com/hats#host
        ' xml:lang='en-us'/>
    <hat displayName='Presenter' name='http://schemas.example.com/hats
        #presenter' xml:lang='en-us'/>
  </hats>
```

```
</presence>
```

3.2 Adding a Hat

Hats are added and removed using Ad-Hoc Commands (XEP-0050) [3].
The following flow shows how to add a hat.

Listing 3: Admin Requests to Add a Hat

```
<iq from='professor@example.edu/office'
    id='fdi3n2b6'
    to='physicsforpoets@courses.example.edu'
    type='set'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           action='execute'
           node='urn:xmpp:hats:commands:don'/>
</iq>
```

Unless an error occurs, the service returns the appropriate form.

Listing 4: Service Returns Form to Admin

```
<iq from='physicsforpoets@courses.example.edu'
    id='fdi3n2b6'
    to='professor@example.edu/office'
    type='result'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='urn:xmpp:hats:commands:don'
           sessionid='A971D19A-2226-4DAD-B261-8D0886B9A026'
           status='executing'>
    <x xmlns='jabber:x:data' type='form'>
      <title>Assigning a Hat</title>
      <instructions>Fill out this form to assign a hat.</instructions>
      <field type='hidden' var='FORM_TYPE'>
        <value>urn:xmpp:hats:commands</value>
      </field>
      <field label='User Address'
             type='jid-single'
             var='accountjid'>
        <required/>
      </field>
      <field label='The role'
             type='list-single'
             var='hat'>
```

[3] XEP-0050: Ad-Hoc Commands <https://xmpp.org/extensions/xep-0050.html>.

```
        <option label='Teacher'><value>http://tech.example.edu/hats#
            Teacher</value></option>
        <option label='Teacher&apos;s␣Assistant'><value>http://tech.
            example.edu/hats#TeacherAssistant</value></option>
        <option label='Test␣Proctor'><value>http://tech.example.edu/
            hats#Proctor</value></option>
      </field>
    </x>
  </command>
</iq>
```

Listing 5: Admin Submits Form

```
<iq from='professor@example.edu/office'
    id='9fens61z'
    to='physicsforpoets@courses.example.edu'
    type='set'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='urn:xmpp:hats:commands:don'
           sessionid='A971D19A-2226-4DAD-B261-8D0886B9A026'>
    <x xmlns='jabber:x:data' type='submit'>
      <field type='hidden' var='FORM_TYPE'>
        <value>urn:xmpp:hats:commands</value>
      </field>
      <field var='accountjid'>
        <value>terry.anderson@example.edu</value>
      </field>
      <field var='hat'>
        <value>http://tech.example.edu/hats#TeacherAssistant</value>
      </field>
    </x>
  </command>
</iq>
```

Listing 6: Service Informs Admin of Completion

```
<iq from='physicsforpoets@courses.example.edu'
    id='9fens61z'
    to='professor@example.edu/office'
    type='result'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='urn:xmpp:hats:commands:don'
           sessionid='A971D19A-2226-4DAD-B261-8D0886B9A026'
           status='completed'/>
</iq>
```

Note: only one hat is added at a time, and the form uses a field of type "list-single" to enforce that logic.

3.3 Removing a Hat

The following flow shows how to remove a hat.

Listing 7: Admin Requests to Remove a Hat

```
<iq from='professor@example.edu/office'
    id='fdi3n2b6'
    to='physicsforpoets@courses.example.edu'
    type='set'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           action='execute'
           node='urn:xmpp:hats:commands:doff'>
    <x xmlns='jabber:x:data' type='submit'>
      <field type='hidden' var='FORM_TYPE'>
        <value>urn:xmpp:hats:commands</value>
      </field>
      <field var='accountjid'>
        <value>terry.anderson@example.edu</value>
      </field>
      <field var='hat'>
        <option label='Teacher&apos;s Assistant'><value>http://tech.
            example.edu/hats#TeacherAssistant</value></option>
      </field>
    </x>
  </command>
</iq>
```

Listing 8: Service Informs Admin of Completion

```
<iq from='physicsforpoets@courses.example.edu'
    id='9fens61z'
    to='professor@example.edu/office'
    type='result'
    xml:lang='en'>
  <command xmlns='http://jabber.org/protocol/commands'
           node='urn:xmpp:hats:commands:doff'
           sessionid='A971D19A-2226-4DAD-B261-8D0886B9A026'
           status='completed'/>
</iq>
```

4 Security Considerations

To follow.

5 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [4].

6 XMPP Registrar Considerations

6.1 Protocol Namespaces

The XMPP Registrar shall add "urn:xmpp:hats:0" to its registry of protocol namespaces.

7 XML Schema

To follow.

8 Acknowledgements

The concepts underlying this specification were first discussed several years ago at an XMPP Summit in Brussels, Belgium. Special thanks to Joe Hildebrand and Ralph Meijer for their contributions to those discussions. Thanks also to Matt Miller, Kevin Smith, and Matthew Wild for their feedback on the written specification.

[4] 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/>.