XEP-0140: Shared Groups

Peter Saint-Andre
mailto:peter@andyet.net
xmpp:stpeter@stpeter.im
https://stpeter.im/

2004-10-27
Version 0.2

Status	Type	Short Name
Retracted	Informational	groups

This document defines a protocol profile for centrally defined and administered roster groups; the protocol described herein has been retracted in favor of XEP-0144: Roster Item Exchange.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

XMPP IM [1] defines a protocol for personal rosters (also known as contact lists). So far all Jabber rosters are personal rosters that are defined by a single user and accessed only by that user. However, in some contexts it would be helpful to centrally define and administer roster groups so that they can be shared among a user population in an organized fashion. This functionality is often called "shared groups".
One context in which shared groups might be useful is the enterprise environment. For example, when Alice is hired by the marketing department of Big Company Enterprises, it makes sense for her to automatically have the other members of the marketing department in her roster the first time she logs in, and for the rest of the marketing department to have Alice in their rosters as soon as her account has been set up. Similarly, when Bob in logistics gets fired, it makes sense for him to disappear from the rosters of everyone else in the logistics department.
This functionality is not limited to the enterprise environment. It could prove quite useful in academic settings, social networking applications, consumer IM services, and anywhere else that it is important to build and manage small communities of users.
Although Roster Item Exchange (XEP-0093) [2] defines a format for sharing roster items between two users and therefore enables one user to send roster items to another user, it does not currently provide a way to share or coordinate a group of roster items in an organized fashion. To make that possible, this document extends XEP-0093 by defining Publish-Subscribe (XEP-0060) [3] as the distribution mechanism, resulting in a basic solution for shared groups over Jabber.

2 Requirements

This document addresses the following use cases:

1. Creating a shared group.
2. Adding a member to the group.
3. Removing a member from the group.

This document does not address the following use cases, which instead are discussed in the Implementation Guidelines section of this document:

- Exchanging presence with members of a shared group.

[1] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.
[2] XEP-0093: Roster Item Exchange <https://xmpp.org/extensions/xep-0093.html>.
[3] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

- Sending a message to all members of a shared group.
- Inviting all members of a shared group to a groupchat room.

3 Use Cases

The following examples show the protocol flow for an administrator to complete the use cases defined above. Naturally, these tasks could be performed just as well by an automated application that is tied to an existing user database (e.g., LDAP).

3.1 Creating a Shared Group

A group is implemented as a pubsub node. If a contact is a member of multiple groups, the contact MUST be added to each pubsub node separately. There is a one-to-one relationship between a group and a node. It is OPTIONAL for the NodeID to include the name of the group (e.g., "groups/Marketing"), although in general it is best not to overload NodeIDs and this is unnecessary given the structure of the groups protocol as described below.

Listing 1: Admin Creates a Group

```
<iq type='set'
    from='bofh@example.com/daygig'
    to='pubsub.example.com'
    id='create1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <create node-id='groups/Marketing/Europe'/>
  </pubsub>
</iq>
```

Listing 2: Service Informs Admin of Success

```
<iq type='result'
    from='pubsub.example.com'
    to='bofh@example.com/daygig'
    id='create1'/>
```

3.2 Adding a Member to the Group

There are two steps to adding a member to a group, which SHOULD be performed in this order:

1. Add new member to subscriber list. (OPTIONAL)
2. Publish member information to node.

Listing 3: Admin Adds Member to Subscriber List

```
<iq type='set'
    from='bofh@example.com/daygig'
    to='pubsub.example.com'
    id='sub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <entities node='groups/Marketing/Europe'>
      <entity jid='alice@example.com'
              affiliation='none'
              subscription='subscribed'/>
    </entities>
  </pubsub>
</iq>
```

Listing 4: Service Informs Admin of Success

```
<iq type='result'
    from='pubsub.example.com'
    to='bofh@example.com/daygig'
    id='sub1'/>
```

(Naturally, a member of a shared group need not be informed of changes to the group, and an entity that is informed of changes to the group need not be a member of the group. However, in most applications a group member will be a pubsub subscriber and vice-versa.)
Now the admin publishes information about the member to the group node.

Listing 5: Admin Publishes Member Information

```
<iq type='set'
    from='bofh@example.com/daygig'
    to='pubsub.example.com'
    id='pub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='groups/Marketing/Europe'>
      <item id='alice@example.com'>
        <x xmlns='jabber:x:roster'
           jid='alice@example.com'
           name='Alice Rosenbaum'>
          <group>Marketing/Europe</group>
        </x>
      </item>
    </publish>
  </pubsub>
</iq>
```

The member information is then delivered to all subscribers:

Listing 6: Member Information is Delivered to All Subscribers

```
<message
    from='pubsub.example.com'
    to='cathy@example.com'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='groups/Marketing/Europe'>
      <item id='alice@example.com'>
        <x xmlns='jabber:x:roster'
           jid='alice@example.com'
           name='Alice Rosenbaum'>
          <group>Marketing/Europe</group>
        </x>
      </item>
    </items>
  </event>
</message>
.
.
.
```

Note: It is the receiving application's responsibility to add the newly-published roster item to the recipient's roster by following the protocols defined in **XMPP IM**. The receiving application SHOULD NOT prompt the recipient regarding whether or not to add the roster item, but if and only the roster item is received via pubsub (i.e., it SHOULD prompt the user when roster items are received from individual users and not via pubsub).

3.3 Removing a Member from the Group

There are two steps to adding a member to a group, which SHOULD be performed in this order:

1. Remove member from subscriber list.
2. Purge member information from node.

Listing 7: Admin Removes Member from Subscriber List

```
<iq type='set'
    from='bofh@example.com/daygig'
    to='pubsub.example.com'
    id='del1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <entities node='groups/Marketing/Europe'>
      <entity jid='alice@example.com'
              affiliation='none'
              subscription='none'/>
    </entities>
  </pubsub>
</iq>
```

Listing 8: Service Informs Admin of Success

```
<iq type='result'
    from='pubsub.example.com'
    to='bofh@example.com/daygig'
    id='del1'/>
```

(As noted, the group member need not be a pubsub subscriber, in which case the foregoing step may not be necessary.)
Now admin can remove the member from the shared group.

Listing 9: Admin Removes Member

```
<iq type='set'
    from='bofh@example.com/daygig'
    to='pubsub.example.com'
    id='purge1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <retract node='groups/Marketing/Europe'>
      <item id='alice@example.com'/>
    </retract>
  </pubsub>
</iq>
```

All remaining subscribers are then informed that the node has been deleted:

Listing 10: Tune Information is Delivered to All Subscribers

```
<message
    from='pubsub.example.com'
    to='cathy@example.com'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <retract node='groups/Marketing/Europe'>
      <item id='alice@example.com'/>
    </retract>
  </event>
</message>
.
.
.
```

4 Implementation Guidelines

4.1 Group Hierarchies

An administrator may wish to define a hierarchy of shared groups (e.g., "Marketing/Europe" and "Marketing/North America"). This can be done using collection nodes as defined in

Section 9 of XEP-0060. The receiving application MAY use Nested Roster Groups (XEP-0083) [4] to define the roster group names.

4.2 Exchanging Presence

Presence is exchanged via the normal mechanisms defined in **XMPP IM**.

4.3 Sending Messages

In order to send a message to all members of a shared group, a group member's sending application (usually an end-user client) SHOULD either send multiple messages or use Extended Stanza Addressing (XEP-0033) [5].

4.4 Sending Groupchat Invites

In order to invite all members of a shared group to a groupchat room, a group member's sending application SHOULD use the mechanisms defined in Multi-User Chat (XEP-0045) [6].

5 Security Considerations

This protocol introduces no security considerations above and beyond those defined in **XEP-0060: Publish-Subscribe** and **XEP-0093: Roster Item Exchange**.

6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [7].

[4] XEP-0083: Nested Roster Groups <https://xmpp.org/extensions/xep-0083.html>.
[5] XEP-0033: Extended Stanza Addressing <https://xmpp.org/extensions/xep-0033.html>.
[6] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[7] 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/>.

7 XMPP Registrar Considerations

This document requires no interaction with the XMPP Registrar [8].

[8] 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/>.