XEP-0083: Nested Roster Groups

Rachel Blackman
mailto:rcb@ceruleanstudios.com
xmpp:sparks@jabber.org

2004-10-11
Version 1.0

Status	Type	Short Name
Active	Informational	nestedgroups

This document defines an XMPP protocol extension that enables nested sub-groups to exist within the Jabber roster, while retaining backwards compatibility and ensuring that the roster remains usable by all clients.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

In many modern instant messaging clients, client authors implement a way to nest contact groups within other contact groups. Usually this is implemented on the client side, since many instant messaging networks do not support nesting contact groups in this manner. The limitation of this system is that if the user changes from one client to another, or even a second installation of the same client, the user loses all of his or her sub-group information. This document aims to solve that problem within Jabber, by providing for a way to store sub-groups on the Jabber roster without breaking existing clients.

2 roster:delimiter Namespace

Jabber already provides a useful method for storing client data on the server using Private XML Storage (XEP-0049) [1]. All we need to do is create a new roster element and a namespace to store the delimiter for nested groups, roster:delimiter. This element MUST contain XML character data that defines a string to be used as a delimiter in the roster groups. [2]
A single-character roster delimiter (e.g., ”/”) would make client implementation easier, but be more limiting to the end-user in terms of choices for naming roster groups, so a string is ideal. Therefore, the delimiter SHOULD contain multiple characters in order to avoid inconveniencing the user, but single-character delimiters MUST be honored by the client. The exception is if the delimiter is a single alphanumeric character (a-z, A-Z, 0-9); in this case compliant clients MUST treat the situation as if nesting were disabled, to avoid malicious use of this element by setting ’e’ or ’m’ or some other common single character as a delimiter.
A compliant client SHOULD ask for the nested delimiter before requesting the user’s roster, in order to know whether or not to parse the roster ’group’ fields accordingly. If there is no delimiter stored, a client MAY set a delimiter but MUST either prompt the user for a delimiter, or use a user-configurable default.

3 Use Cases

Use cases for this extension are straightforward, and are shown below as examples.

3.1 Querying for the delimiter

All compliant clients SHOULD query for an existing delimiter at login.

[1] XEP-0049: Private XML Storage <https://xmpp.org/extensions/xep-0049.html>.
[2] If the element does not contain XML character data, a compliant client SHOULD assume that nested groups are disabled for the user’s account.

Listing 1: Querying for the Delimiter

```
CLIENT:
<iq type='get'
    id='1'>
  <query xmlns='jabber:iq:private'>
    <roster xmlns='roster:delimiter'/>
  </query>
</iq>

SERVER:
<iq type='result'
    id='1'
    from='bill@shakespeare.lit/Globe'
    to='bill@shakespeare.lit/Globe'>
  <query xmlns='jabber:iq:private'>
    <roster xmlns='roster:delimiter'>::</roster>
  </query>
</iq>
```

3.2 Retrieving the Roster

Now that the client has a delimiter, we can retrieve and parse the roster properly.

Listing 2: Retrieving a Roster with Nested Groups

```
CLIENT:
<iq type='get'
    id='2'>
  <query xmlns='jabber:iq:roster'/>
</iq>

SERVER:
<iq type='result'
    to='bill@shakespeare.lit/Globe'
    id='2'>
  <query xmlns='jabber:iq:roster'>
    <item jid='bottom@athens.gr' subscription='both'>
      <group>Midsummer::Actors</group>
    </item>
    <item jid='quince@athens.gr' subscription='both'>
      <group>Midsummer::Actors</group>
    </item>
    <item jid='snug@athens.gr' subscription='both'>
      <group>Midsummer::Actors</group>
    </item>
    <item jid='theseus@athens.gr' subscription='both'>
      <group>Midsummer::Royalty</group>
    </item>
```

```
    <item jid='hippolyta@athens.gr' subscription='both'>
      <group>Midsummer::Royalty</group>
    </item>
    <item jid='robin@faeries.underhill.org' subscription='both'>
      <group>Midsummer</group>
    </item>
    <item jid='hamlet@denmark.net' subscription='both'>
      <group>Hamlet</group>
    </item>
    <item jid='gertrude@denmark.net' subscription='both'>
      <group>Hamlet</group>
    </item>
  </query>
</iq>
```

Bottom, Quince and Snug should be grouped together in a group 'Actors' underneath the group 'Midsummer'. Theseus and Hippolyta should be grouped together in a group 'Royalty' under 'Midsummer'. Robin Goodfellow, meanwhile, being in a class unto himself, is a plain contact under the 'Midsummer' group rather than in an actual sub-group. The group Hamlet, containing only one melancholy prince and his mother, contains no sub-groups at all.
As should be apparent, a client which does not support the delimiter will instead create a separate group -- such as 'Midsummer::Actors' -- and thus will still have each set of contacts grouped with the other appropriate contacts.

4 Security Considerations

There are no security features or concerns related to this proposal above and beyond those specified for roster management in XMPP IM [3] and for private XML storage in XEP-0049.

5 IANA Considerations

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

[3] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.

[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/>.

6 XMPP Registrar Considerations

6.1 Protocol Namespaces

The XMPP Registrar [5] includes 'roster:delimiter' in its registry of protocol namespaces.

7 Schema

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='roster:delimiter'
    xmlns='roster:delimiter'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0083: http://www.xmpp.org/extensions/xep-0083.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='roster' type='xs:string'/>

</xs:schema>
```

[5] 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/>.