XEP-0209: Metacontacts

Kevin Smith
mailto:kevin@kismith.co.uk
xmpp:kevin@doomsong.co.uk

Remko Tronçon
http://el-tramo.be/

Yann Le Boulanger
mailto:asterix@lagaule.org
xmpp:asterix@jabber.lagaule.org

2007-04-10
Version 0.1

Status	Type	Short Name
Deferred	Standards Track	NOT YET ASSIGNED

This document specifies an XMPP protocol extension for defining metacontacts and grouping member JIDs.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2024 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the ”Specification”), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an ”AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

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 **1**

2 Requirements **1**

3 Overview **1**

4 Use Cases **2**
- 4.1 Retrieving the metacontact data . . . 2
- 4.2 Storing the metacontact data . . . 3

5 Implementation Notes **3**
- 5.1 Creating a metacontact . . . 3
- 5.2 Removing a metacontact . . . 4
- 5.3 Uniqueness of order within a metacontact . . . 4

6 Security Considerations **4**

7 IANA Considerations **4**

8 XMPP Registrar Considerations **4**

9 XML Schema **5**

1 Introduction

It is often the case that a user will have multiple representations of a single contact, either within one account (as is often the case with contacts using legacy systems through transports) or across several accounts (particularly where a user or contact has separate work and home accounts). As these are different representations of the same logical entity, this XEP provides a method for binding them together into a single meta-contact.

2 Requirements

The authors have designed the metacontacts protocol with the following requirements in mind:

- It MUST provide a method of consolidating multiple roster contacts representing the same logical entity both within an account and, if the user utilises multiple accounts, between accounts.
- In the case of meta-contacts spanning multiple accounts, the meta-contact data must be resilient to the failure (or simply the absence) of any combination of these accounts. Particularly it MUST NOT rely upon the private storage of one account to store data for other accounts.
- It MUST allow a user to order the contacts within a meta-contact in a hierarchy which clients are able to use to determine which should be the default recipient of messages where more than one is available at any time but MUST NOT require the enforcement of a hierarchy.

3 Overview

The metacontact storage is achieved through the use of private data storage. In order to achieve the resilience described above, the private storage of each account is used to store the metacontact membership of Jids in the roster of that account. The metacontacts are stored within private data storage of each account simply as an unordered collection of meta tags.

Listing 1: A metacontact tag

```
<meta jid='romeo@montague.net' tag='d93nov' order='0'>
```

In this example, the 'jid' specifies that the roster entry 'romeo@montague.net' is a member of a metacontact. The 'tag' provides a key for a metacontact; in this example all metacontacts with a tag of 'd93nov' (across all accounts) refer to the same entity. The 'order' denotes the priority of this Jid over other Jids within the metacontact, with it being preferable to use Jids with higher priority (this is roughly analogous to the 'priority' on presence stanzas when a

Jid has multiple online resources in XMPP).

4 Use Cases

Below are example of setting and retrieving metacontacts for an account. When using metacontacts across multiple accounts, the steps are identical and the 'tag' attributes and used across accounts (that is: when the same tag is used for multiple contacts, all entries with the tag are merged into a single metacontact whether they reside on the same of different accounts).

4.1 Retrieving the metacontact data

Upon login, a client will need to know the current state of the metacontact structure and so SHOULD request the state for each account.

Listing 2: Requesting metacontacts

```
<iq type='get' id='get1'>
  <query xmlns='jabber:iq:private'>
    <storage xmlns='storage:metacontacts'/>
  </query>
</iq>
```

The result of the query will list the metacontact membership for roster entries belonging to the queried account.

Listing 3: Server returns metacontact data

```
<iq type='result' id='get1'>
  <query xmlns='jabber:iq:private'>
    <storage xmlns='storage:metacontacts'>
      <meta jid='mike.bolton@raplovers.org' tag='ae18f2' order='1'/>
      <meta jid='tom@jump-to-conclusions.com' tag='82a1a5' order='1'/>
      <meta jid='samir@initech.com' tag='283b94' order='2'/>
      <meta jid='mike@initech.com' tag='ae18f2' order='2'/>
    </storage>
  </query>
</iq>
```

The meta tags each represent a contact which is a member of a metacontact; as such it is likely that some roster entries for an account will not have corresponding meta tags (if they are not members of a metacontact). Each 'meta' tag MUST have a 'jid' attribute and a 'tag' attribute, an optional 'order' attribute MAY be included.
The value of the 'jid' attribute MUST be the bare JID of a contact which is a member of the

described metacontact and any jid MUST NOT be specified in this manner as a member of more than one metacontact within an account. A JID specified as a member of a metacontact for an account SHOULD be a member of the roster for that account.
The value of the 'tag' is used as a non-human readable unique identifier for a metacontact. In the above example, there are three metacontacts; those with tags of '82a1a5' and '283b94' each have only one contact on this account, while the metacontact with tag tag='ae18f2' contains two contacts.
The value of the 'order' attribute is used to suggest preference of some contacts over others within a metacontact; those contacts with a higher order are preferable to those with a lower order. In this example, 'mike@initech.com' is considered preferable to 'mike.bolton@raplovers.org' when communicating with the metacontact with tag 'ae18f2', due to their order of 2 and 1 respectively.
Note: It is entirely possible for the query result for an account to indicate that, in isolation from other accounts, there exist metacontacts which contain only a single member contact, as is the case in the above example. The client MUST NOT discard such data, as other accounts belonging to the user (which the client need not have details of) may also have members of this metacontact.

4.2 Storing the metacontact data

```
<iq type='set' id='set2'>
  <query xmlns='jabber:iq:private'>
    <storage xmlns='storage:metacontacts'>
      <meta jid='mike.bolton@raplovers.org' tag='ae18f2' order='1'/>
      <meta jid='tom@jump-to-conclusions.com' tag='82a1a5' order='1'/>
      <meta jid='samir@initech.com' tag='283b94' order='2'/>
      <meta jid='milton@hawaii.com' tag='492ab2' order='2'/>
      <meta jid='joanna@kung-fu.org' tag='9248cc' order='1'/>
    </storage>
  </query>
</iq>
```

Listing 4: Server returns

```
<iq type='result' id='set2'/>
```

5 Implementation Notes

5.1 Creating a metacontact

Creation of a metacontact is uncomplicated; the simple addition of meta tag with a common tag results in a new metacontact.

5.2 Removing a metacontact

Similarly, to remove a metacontact all that is required is to remove the meta tags which contribute to the metacontact.

5.3 Uniqueness of order within a metacontact

Although it is unavoidable that multiple contacts within a metacontact MAY have the same order (due to potentially unavailable information from other accounts), clients SHOULD NOT apply the same order to multiple members of the same metacontact where it is possible to avoid it. If multiple members of a metacontact have the same order, the behaviour is dependent upon the client; it MAY apply rules itself to determine which member to communicate with (based upon presence, recent activity or other methods) it MAY present the user with the option to sort the members such that the orders are again unique, or it MAY employ another appropriate action.
As the order attribute is optional, clients need a method for determining which member contact to use where the metacontact consists entirely of unordered members. When ordered and unordered members are present, unordered members SHOULD be considered to have the lowest order.

6 Security Considerations

Security considerations related to private XML storage are described in XEP-0049; this XEP introduces no additional concerns.

7 IANA Considerations

No interaction with the Internet Assigned Numbers Authority (IANA) [1] is required as a result of this XEP.

8 XMPP Registrar Considerations

No namespaces or parameters need to be registered with the XMPP Registrar [2] as a result of this XEP.

[1] 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/>.

[2] 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/>.

9 XML Schema

```
<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='storage:metacontacts'
    xmlns='storage:metacontacts'
    elementFormDefault='qualified'>

  <xs:element name='metacontacts'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='meta' minOccurs='0' maxOccurs='unbounded'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='meta'>
    <xs:complexType>
      <xs:attribute name='jid' type='xs:string' use='required'/>
      <xs:attribute name='tag' type='xs:string' use='required'/>
      <xs:attribute name='order' type='xs:xs:unsignedInt' use='
          optional'/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```