XEP-0144: Roster Item Exchange

Peter Saint-Andre
mailto:stpeter@jabber.org
xmpp:stpeter@jabber.org
https://stpeter.im/

2005-08-26
Version 1.0

Status	Type	Short Name
Draft	Standards Track	rosterx

This specification defines an XMPP protocol extension for exchanging contact list items, including the ability to suggest whether the item is to be added, deleted, or modified in the contact list of the recipient, as well as the suggested roster group for the item.

Legal

Copyright

Permissions

Warranty

Liability

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 <http://xmpp.org/about-xmpp/xsf/xsf-ipr-policy/> or obtained by writing to XMPP Standards Foundation, 1899 Wynkoop Street, Suite 600, Denver, CO 80202 USA).

Contents

1 Introduction

The Jabber protocols have long included a method for sending roster items from one entity to another, making use of the 'jabber:x:roster' namespace. Because this protocol extension was not required by RFC 2779 [1], it was removed from XMPP IM [2] and documented for historical purposes in Roster Item Exchange [3]. However, since that time discussions in the Standards SIG [4] have revealed that it would be helpful to use roster item exchange in the problem spaces of "shared groups" (e.g., predefined roster groups used within an organization) and roster synchronization (e.g., keeping a Jabber roster in sync with a contact list on a legacy IM service). These problem spaces require a slightly more sophisticated kind of roster item exchange than was documented in XEP-0093, specifically the ability to indicate whether a roster item is to be added, deleted, or modified. Therefore this document redefines roster item exchange to provide this functionality in a way that is backwards-compatible with existing implementations, albeit using a modern namespace URI of 'http://jabber.org/protocol/rosterx' rather than the old 'jabber:x:roster' namespace name. Further specifications will define how to solve the problems of shared groups and roster synchronization using the protocol defined herein.

2 Requirements

XEP-0093 did not define the requirements for roster item exchange. This section remedies that oversight.
Roster item exchange meets the following requirements:

1. Enable an entity to send one or more roster items to another entity, with the suggestion that the roster item(s) be added to the recipient's roster.
2. Enable an entity to send one or more roster items to another entity, with the suggestion that the roster item(s) be deleted from the recipient's roster.
3. Enable an entity to send one or more roster items to another entity, with the suggestion that the roster item(s) be modified in the recipient's roster.

This document deliberately speaks of rosters and roster items, not presence subscriptions. Although rosters and subscriptions are closely connected (as explained in RFC 3921), they are not identical. The protocol defined herein enables an entity to suggest that another entity might want to add, delete, or modify roster items only, and does not dictate the suggested

[1] RFC 2779: A Model for Presence and Instant Messaging <http://tools.ietf.org/html/rfc2779>.
[2] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.
[3] XEP-0093: Roster Item Exchange <http://xmpp.org/extensions/xep-0093.html>.
[4] The Standards SIG is a standing Special Interest Group devoted to development of XMPP Extension Protocols. The discussion list of the Standards SIG is the primary venue for discussion of XMPP protocol extensions, as well as for announcements by the XMPP Extensions Editor and XMPP Registrar. To subscribe to the list or view the list archives, visit <http://mail.jabber.org/mailman/listinfo/standards/>.

presence subscription state associated with those roster items. This is intentional.

3 Use Cases

3.1 Suggesting Roster Item Addition

In order to programatically suggest that the receiving entity should add one or more items to its roster, the sending entity MUST send a <message/> or <iq/> stanza containing an <x/> element qualified by the 'http://jabber.org/protocol/rosterx' namespace (see Recommended Stanza Type regarding when to use <message/> and when to use <iq/>); the <x/> element in turn MUST contain one or more <item/> child elements, each of which SHOULD possess an 'action' attribute whose value is "add" [5], MUST possess a 'jid' attribute that specifies the JabberID of the item to be added, MAY possess a 'name' attribute that specifies a natural-language name or nickname for the item, and MAY contain one or more <group/> elements specifying roster groups into which to place the item. If a <message/> stanza is sent, it MAY contain a <body/> element but SHOULD NOT contain any other child elements. Here is an example:

Listing 1: Suggesting Addition

```
<message from='horatio@denmark.lit' to='hamlet@denmark.lit'>
  <body>Some visitors, m'lord!</body>
  <x xmlns='http://jabber.org/protocol/rosterx'>
    <item action='add'
          jid='rosencrantz@denmark.lit'
          name='Rosencrantz'>
      <group>Visitors</group>
    </item>
    <item action='add'
          jid='guildenstern@denmark.lit'
          name='Guildenstern'>
      <group>Visitors</group>
    </item>
  </x>
</message>
```

In determining how to handle any given roster item whose 'action' attribute has a value of "add" (either explicitly or as the default value), the receiving application SHOULD proceed as follows:

1. If the item already exists in the roster and the item is in the specified group (or no group is specified), the receiving application MUST NOT prompt a human user for approval regarding that item and MUST NOT add that item to the roster.

[5] The default value of the 'action' attribute is "add"; therefore, if the 'action' attribute is not included or the receiving application does not understand the 'action' attribute, the receiving application MUST treat the item as if the value were "add".

2. If the item does not already exist in the roster, the receiving application SHOULD prompt a human user for approval regarding that item and, if approval is granted, MUST add that item to the roster.

3. If the item already exists in the roster but not in the specified group, the receiving application MAY prompt the user for approval and SHOULD edit the existing item so that will also belong to the specified group (in addition to the existing group, if any).

If the roster item addition stanza will result in adding the item to the roster, the receiving application MUST (either with approval by a human user or automatically subject to configuration) send a roster set to the user's server containing the new item as described in RFC 3921. After completing the roster set, the receiving application SHOULD also send a <presence/> stanza of type "subscribe" to the JID of the new item.
For a description of the recommended application behavior when a roster item addition stanza actually results in editing of an existing roster item, refer to the Suggesting Roster Item Modification section of this document.

3.2 Suggesting Roster Item Deletion

In order to programatically suggest that the receiving entity should delete one or more items from its roster, the sending entity MUST send a <message/> or <iq/> stanza containing an <x/> element qualified by the 'http://jabber.org/protocol/rosterx' namespace; the <x/> element in turn MUST contain one or more <item/> child elements, each of which MUST possess an 'action' attribute whose value is "delete", MUST possess a 'jid' attribute that specifies the JabberID of the item to be deleted, MAY possess a 'name' attribute that specifies a natural-language name or nickname for the item, and MAY contain one or more <group/> elements specifying roster groups for the item. If a <message/> stanza is sent, it MAY contain a <body/> element but SHOULD NOT contain any other child elements. Here is an example:

Listing 2: Suggesting Deletion

```
<message from='horatio@denmark.lit' to='hamlet@denmark.lit'>
  <x xmlns='http://jabber.org/protocol/rosterx'>
    <item action='delete'
          jid='rosencrantz@denmark'
          name='Rosencrantz'>
      <group>Visitors</group>
    </item>
    <item action='delete'
          jid='guildenstern@denmark'
          name='Guildenstern'>
      <group>Visitors</group>
    </item>
  </x>
</message>
```

In determining how to handle any given roster item whose 'action' attribute has a value of "delete", the receiving application SHOULD proceed as follows:

1. If the item does not exist in the roster, the receiving application MUST NOT prompt a human user for approval regarding that item and MUST NOT delete that item from the roster.
2. If the item exists in the roster but not in the specified group, the receiving application MUST NOT prompt the user for approval and MUST NOT delete the existing item.
3. If the item exists in the roster and is in both the specified group and another group, the receiving application MAY prompt the user for approval and SHOULD edit the existing item so that it no longer belongs to the specified group.

If the item is to be deleted, the receiving application SHOULD remove the item from the roster by sending a roster set to the user's server with the 'subscription' attribute set to a value of "remove" as described in RFC 3921, since this results in generation of the appropriate <presence/> stanzas by the user's server.

3.3 Suggesting Roster Item Modification

In order to programatically suggest that the receiving entity should modify one or more items from its roster, the sending entity MUST send a <message/> or <iq/> stanza containing an <x/> element qualified by the 'http://jabber.org/protocol/rosterx' namespace; the <x/> element in turn MUST contain one or more <item/> child elements, each of which MUST possess an 'action' attribute whose value is "modify", MUST possess a 'jid' attribute that specifies the JabberID of the item to be modified, MAY possess a 'name' attribute that specifies a natural-language name or nickname for the item, and MAY contain one or more <group/> elements specifying roster groups into which to place the item. If a <message/> stanza is sent, it MAY contain a <body/> element but SHOULD NOT contain any other child elements. Here is an example:

Listing 3: Suggesting Modification

```
<message from='horatio@denmark.lit' to='hamlet@denmark.lit'>
  <x xmlns='http://jabber.org/protocol/rosterx'>
    <item action='modify'
          jid='rosencrantz@denmark.lit'
          name='Rosencrantz'>
      <group>Retinue</group>
    </item>
    <item action='modify'
          jid='guildenstern@denmark.lit'
          name='Guildenstern'>
      <group>Retinue</group>
```

```
    </item>
  </x>
</message>
```

In determining how to handle any given roster item whose 'action' attribute has a value of "modify", the receiving application SHOULD proceed as follows:

1. If the item does not exist in the roster, the receiving application MUST NOT prompt a human user for approval regarding that item and MUST NOT add that item to the roster.
2. If the item exists in the roster and the modification results in a change of group only, the receiving application MAY prompt the user for approval and SHOULD move the item to the specified group.
3. If the item exists in the roster and the modification results in adding the item to a new group in addition to its existing group, the receiving application MAY prompt the user for approval and SHOULD add the item to the specified group.
4. If the item exists in the roster and the modification results in a change of name only, the receiving application MAY prompt the user for approval and SHOULD modify the name to that specified in the modification suggestion.

If a roster item addition, deletion, or modification stanza will result in editing of an existing item in the roster, the receiving application MUST (either with approval by a human user or automatically subject to configuration) send a roster set to the user's server with no changes to the 'subscription' attribute but rather with appropriate changes to the value of 'name' attribute or the <group/> child element or elements, as described in RFC 3921.

4 Service Discovery

In order to determine whether a receiving entity supports the protocol defined herein, the sending entity SHOULD use Service Discovery [6] but MAY depend on the "profile" of Service Discovery defined in Entity Capabilities [7]. If an entity supports roster item exchange, it MUST (subject to appropriate security considerations as described under Advertising Support) include <feature var='http://jabber.org/protocol/rosterx'/> in its responses to disco#info queries. Thus a sending entity can discover if a receiving entity supports the protocol defined herein by sending an IQ request of the following form:

Listing 4: Sending Entity Queries for Support

```
<iq from='horatio@denmark.lit/castle'
    to='hamlet@denmark.lit/throne'
```

[6] XEP-0030: Service Discovery <http://xmpp.org/extensions/xep-0030.html>.
[7] XEP-0115: Entity Capabilities <http://xmpp.org/extensions/xep-0115.html>.

```
    type='get'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

The receiving entity then indicates its support:

Listing 5: Receiving Entity Advertises Support

```
<iq from='hamlet@denmark.lit/throne'
    to='horatio@denmark.lit/castle'
    type='get'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature var='http://jabber.org/protocol/rosterx'/>
    ...
  </query>
</iq>
```

5 Recommended Stanza Type

If the sending entity has knowledge (e.g., via presence or an active chat conversation) that the receiving entity is online and available, it SHOULD: [8]

1. Discover if the receiving entity supports the protocol defined herein (see the Service Discovery section of this document).
2. If so, send its roster item exchange stanza to a particular resource (user@host/resource) of the receiving entity using an <iq/> stanza rather than a <message/> stanza.

If the sending entity does not know that the receiving entity is online and available, it MUST send a <message/> stanza to the receiving entity's "bare JID" (user@host) rather than an <iq/> stanza to a particular resource.

5.1 IQ Semantics

If the sending entity uses <iq/> stanzas to communicate its roster item exchange suggestions, the receiving entity MUST adhere to the IQ semantics defined in XMPP Core [9]. Specifically:

[8] If the receiving entity has more than one available resource, the sending application SHOULD communicate with the "most available" resource according its best estimation (e.g., the resource with the highest priority).

[9] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

1. If the receiving entity successfully processes the suggested action(s) (which may include ignoring certain suggestions), the receiving entity MUST return an empty IQ result to the sending entity.

2. If the receiving entity does not understand the roster item exchange namespace, the receiving entity MUST return an error to the sending entity, which error SHOULD be <service-unavailable/>.

3. If the receiving entity will not process the suggested action(s) because the receiving entity is not registered with the sending entity, the receiving entity MUST return an error to the sending entity, which error SHOULD be <registration-required/>.

4. If the receiving entity will not process the suggested action(s) because the sending entity is not in the receiving entity's roster, the receiving entity MUST return an error to the sending entity, which error SHOULD be <not-authorized/>.

5. If the receiving entity will not process the suggested action(s) because the sending entity is not trusted (see Trusted Entities), the receiving entity MUST return an error to the sending entity, which error SHOULD be <forbidden/>.

Naturally, other IQ errors may be more appropriate; however, if the receiving entity will not or cannot process the suggested action(s), it MUST return an error to the sending entity.

6 Business Rules

1. The sending entity or sending application MUST NOT send additions, deletions, and modifications in the same <x/> element and <message/> or <iq/> stanza; instead, it SHOULD send separate stanzas for the additions, deletions, and modifications.

2. If approval is required or recommended regarding more than one item suggested by the sending entity, the receiving entity SHOULD present all of the items for approval at the same time or in the same interface.

3. If the sending entity is in some sense "trusted" (see Trusted Entities), then the receiving application MAY skip the approval steps described above.

4. The receiving application SHOULD NOT accept an unreasonable number of roster items from any one sending entity at one time. Unfortunately, it can be difficult to determine how many roster items count as "unreasonable". For example, when a user registers with a gateway, it is possible that the initial set of roster items may be quite large (however, note that most existing consumer IM services enforce a limit of 100 to 150 items in their contact lists). Users who have newly registered with or been newly

created on a server (e.g., within an organization) may also receive a large set of initial roster items in order to sync up with shared groups established on the server. However, after such initialization, the subsequent roster item sets should be much smaller. In any case, sets of more than 150 or 200 roster items SHOULD be treated with suspicion, and entities that repeatedly send such sets SHOULD NOT be trusted.

7 Types of Sending Entities

The foregoing protocol description speaks only of "sending entities" and does not differentiate between different types of sending entities. However, it is envisioned that roster items will be sent to receiving entities by three different kinds of senders:

1. Users of Jabber clients.
2. Client proxy gateways.
3. Shared group services.

These are described more completely below.

7.1 Jabber Users

Roster item exchange as developed within the early Jabber community and documented in XEP-0093 was used to send a roster item from one user to another in a manner more structured than simply typing a third party's JID in a chat window. This usage is still encouraged. However, if the sender is a human user and/or the sending application has a primary Service Discovery category of "client" (e.g., a bot) [10], the sending application SHOULD NOT specify an 'action' attribute other than "add", the receiving application MAY ignore values of the 'action' attribute other than "add", and the receiving application MUST prompt a human user regarding whether to add the suggested item or items to the user's roster.

7.2 Gateways

The nature of client proxy gateways (i.e., entities with a service discovery category of "gateway") is specified more fully in Gateway Interaction [11]. Herein we describe how such gateways should use roster item exchange, and how receiving applications should treat roster items received from such gateways.
In order to make use of a gateway's protocol translation service, a user MUST first register with the gateway. If the gateway advertises support for a service discovery feature of

[10] See <http://www.xmpp.org/registrar/disco-categories.html#client>.
[11] XEP-0100: Gateway Interaction <http://xmpp.org/extensions/xep-0100.html>.

'http://jabber.org/protocol/rosterx', then the user's client SHOULD expect that it may receive roster item suggestions from the gateway. In order to maintain synchronization between the user's contact list on a legacy IM service and the user's Jabber roster, the gateway SHOULD send roster items with an 'action' attribute of "add", "delete", or "modify" as appropriate, and the receiving application SHOULD process such roster item suggestions. Such processing MAY occur automatically (i.e., without the user's approval of each roster item or batch of roster items) if and only if the receiving application has explicitly informed the user that it will automatically process roster items from the gateway. Furthermore, the receiving application SHOULD periodically verify automatic processing with the user (e.g., once per session in which the gateway sends roster item suggestions to the user).

7.3 Group Services

There is a third category of entities that might initiate roster item exchanges, which we label a "group service" and identify by a Service Discovery category of "directory" and type of "group". A group service enables an administrator to centrally define and administer roster groups so that they can be shared among a user population in an organized fashion. Such a service could prove useful in enterprise environments [12] and other settings where it is beneficial to synchronize rosters across individuals (e.g., schools, social networking applications, consumer IM services, and anywhere else that it is important to build and manage small communities of users).

In some contexts, an IM server could function as a group service (e.g., if there is a single server deployed on a small company intranet); in other contexts, it may make more sense to deploy a standalone group service (e.g., in a larger or more heterogeneous environment with users on multiple servers). In both cases, the group service MUST advertise a service discovery identity of "directory/group" and SHOULD use the protocol specified herein to communicate changes ("add", "delete", and "modify") to the relevant shared groups; in addition, a user MUST first register with the service (either over Jabber via In-Band Registration [13] or out of band, e.g., via the web) or be otherwise provisioned to use the service (e.g., by a system administrator) before accepting roster item suggestions from the service.

If the user has registered with a group service or been otherwise provisioned to use a group service, the receiving application SHOULD process roster item suggestions received from the service. Such processing MAY occur automatically (i.e., without the user's approval of each roster item or batch of roster items) if and only if the receiving application has explicitly informed the user that it will automatically process roster items from the service. Furthermore, the receiving application SHOULD periodically verify automatic processing with the user (e.g., once per session in which the service sends roster item suggestions to the user).

[12] 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.

[13] XEP-0077: In-Band Registration <http://xmpp.org/extensions/xep-0077.html>.

8 Security Considerations

8.1 Trusted Entities

A principal (user) or receiving application MAY establish a list of trusted entities from which roster item exchanges are processed automatically, i.e., without explicit approval by a human user. In order to avoid corruption of the roster, it is STRONGLY RECOMMENDED that such trusted entities be limited to gateways and group services as defined above. In addition, the receiving application SHOULD periodically verify such automatic processing with the principal, e.g., once per session in which the trusted entity sends roster item suggestions to the user.

8.2 Denial of Service

A sending entity could effectively deny service to the receiving entity by rapidly and repeatedly sending (1) alternating add and delete suggestions or (2) modify suggestions, thus invoking throttling mechanisms enforced by the receiving entity's server. The receiving application SHOULD guard against this by monitoring roster item exchanges received from each sending entity and refusing or ignoring roster item exchanges from offending entities (e.g., by adding such entities to a list of distrusted entities).

8.3 Advertising Support

A receiving application MAY refuse to advertise its support for the roster item exchange protocol (see the Service Discovery section of this document) to entities that that are (1) not explicitly trusted or (2) explicitly distrusted.

9 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [14].

[14] 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/>.

10 XMPP Registrar Considerations

10.1 Protocol Namespaces

The XMPP Registrar [15] includes 'http://jabber.org/protocol/rosterx' in its registry of protocol namespaces.

10.2 Service Discovery Identities

The XMPP Registrar includes a Service Discovery type of "group" under the "directory" category in its registry of service discovery identities.

11 XML Schema

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='http://jabber.org/protocol/rosterx'
    xmlns='http://jabber.org/protocol/rosterx'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0144: http://www.xmpp.org/extensions/xep-0144.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='x'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='item' minOccurs='1' maxOccurs='unbounded'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='item'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='group' type='xs:string' minOccurs='0'
            maxOccurs='unbounded'/>
```

[15] 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 <http://xmpp.org/registrar/>.

```
      </xs:sequence>
      <xs:attribute name='action' use='optional' default='add'>
        <xs:simpleType>
          <xs:restriction base='xs:NCName'>
            <xs:enumeration value='add'/>
            <xs:enumeration value='delete'/>
            <xs:enumeration value='modify'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name='jid' type='xs:string' use='required'/>
      <xs:attribute name='name' type='xs:string' use='optional'/>
    </xs:complexType>
  </xs:element>

</xs:schema>
```