XEP-0283: Moved

Matthew Wild
mailto:mwild1@gmail.com
xmpp:me@matthewwild.co.uk

2021-07-20
Version 0.2.0

Status	Type	Short Name
Experimental	Standards Track	moved

This document defines an XMPP protocol extension that enables a user to inform its contacts about a change in JID.

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 Glossary 1

4 Use Cases 1
- 4.1 User publishes moved statement . . . 2
- 4.2 User notifies contacts of move . . . 2
- 4.3 Contact receives subscription request with moved notification . . . 2
- 4.4 Server-side processing of inbound moved notification . . . 4

5 Implementation Notes 6
- 5.1 Lifetime of moved statement . . . 6
- 5.2 Alternative verification methods . . . 6

6 Security Considerations 6
- 6.1 User considerations . . . 6
- 6.2 Contact considerations . . . 7

7 IANA Considerations 7

8 XMPP Registrar Considerations 7

9 Design Considerations 8
- 9.1 Verification methods . . . 8

10 XML Schema 8

11 Acknowledgements 8

1 Introduction

There are a variety of reasons why a user may wish to change their JID. For example, a surname change because of marriage or simply an easier JID to remember. Another common reason is that the provider goes out of service (with a notice).
This XEP defines an approach for communicating that your JID has moved to a new JID, extending the existing subscription protocol documented in XMPP IM [1]. The steps outlined here may be done either through a client or automated by a server.

2 Requirements

This document aims to satisfy the following requirements:

- Users should be able to notify their contacts of a change of address.
- Contacts should be able to verify that such a notification is legitimate, to prevent malicious actors from spoofing notifications.
- It should be possible for a contact's server to automatically update the contact's roster for seamless migrations.
- In the absence of a contact's support for this protocol, it should fall back to a simple manual subscription approval.

3 Glossary

Moved notification An XML <moved/> element sent to a contact to inform them that the user is moving to a new address.

Moved statement An XML <moved/> element published by the user on their old account. It is used by contacts to verify that moved notifications are genuine.

4 Use Cases

We start with the situation where the user, let's call them Juliet, has two accounts - her original account juliet@im.example.net and a shiny new account on her personal domain, juliet@capulet.example.
Juliet would like to migrate all her data and contacts to her new account, with minimal disruption.

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

4.1 User publishes moved statement

Before notifying contacts of the move, Juliet must connect to her old account and publish a "statement" that the account is no longer in use. This statement includes the address of her new account.
The statement should be posted to a PEP node with the name 'urn:xmpp:moved:1'. The payload should be a <moved/> element in the 'urn:xmpp:moved:1' namespace. This element should in turn contain a single <new-jid/> element with the user's new JID as its text content.

Listing 1: Juliet's client publishes a moved statement on her old account

```
<iq from='juliet@im.example.net/VR0sAGae'
    type='set'
    id='pub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:moved:1'>
      <item id='current'>
        <moved xmlns='urn:xmpp:moved:1'>
          <new-jid>juliet@capulet.example</new-jid>
        </moved>
      </item>
    </publish>
  </pubsub>
</iq>
```

4.2 User notifies contacts of move

After publishing a moved statement on her old account, Juliet proceeds to notify her contacts about the move.
Juliet connects to her new account, and sends a subscription request to each of her contacts. These subscription requests MUST contain a <moved/> element in the 'urn:xmpp:moved:1' namespace. This element contains a single <old-jid/> element with the old JID as its text content.>

Listing 2: Juliet sends a subscription request to Romeo from her new account

```
<presence type='subscribe' to='romeo@montague.example' id='sub1'>
  <moved xmlns='urn:xmpp:moved:1'>
    <old-jid>juliet@im.example.net</old-jid>
  </moved>
</presence>
```

4.3 Contact receives subscription request with moved notification

Juliet's contact, Romeo, receives the subscription request from Juliet's new JID. At this point Romeo has not verified that the new account actually belongs to Juliet, and MUST perform

such verification before acting on the request any differently to usual.
If the value of <old-jid/> is not in the roster with an approved incoming subscription ('from' or 'both'), the <moved/> element MUST be ignored entirely.
To verify the request, Romeo makes a request to the JID specified in <old-jid/> (which MUST be a bare JID) to fetch a published move statement.

Listing 3: Romeo requests a moved statement from Juliet's old account

```
<iq type='get'
    from='romeo@capulet.example/laptop'
    to='juliet@im.example.net'
    id='83hKgF'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='urn:xmpp:moved:1'>
      <item id='current'/>
    </items>
  </pubsub>
</iq>
```

On success, Juliet's server will return the moved statement that Juliet published.

Listing 4: Juliet's old server returns the published moved statement to Romeo

```
<iq type='result'
    from='juliet@im.example.net'
    to='romeo@capulet.example/laptop'
    id='83hKgF'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='urn:xmpp:moved:1'>
      <item id='current'>
        <moved xmlns='urn:xmpp:moved:1'>
          <new-jid>juliet@capulet.example</new-jid>
        </moved>
      </item>
    </items>
  </pubsub>
</iq>
```

Romeo MUST now verify that the received subscription request was from the same bare JID contained in the <new-jid/> element in the moved statement. If the JIDs do not match, or if there was an error fetching the moved statement (except for "gone" - see note below), the <moved/> element in the incoming subscription request MUST be ignored entirely.
Note: As a special case, if the attempt to retrieve the moved statement results in an error with the <gone/> condition as defined in RFC 6120, and that <gone/> element contains a valid XMPP URI (e.g. xmpp:user@example.com), then the error response MUST be handled equivalent to a <moved/> statement containing a <new-jid/> element with the JID provided in the URI (e.g. user@example.com).

Upon successful verification, Romeo's client may present an appropriate user interface, informing about Juliet's change of address, and a prompt to accept the subscription request from the new address. On the user's approval of such a subscription request, the client will typically want to also send a subscription request to the contact's new JID to establish a mutual subscription.
Due to the potential for races between multiple clients connected to the same account, it is NOT RECOMMENDED for a client to automatically act upon migration notifications, but instead await manual interaction from the user. As with any inbound subscription request it SHOULD pay attention to roster pushes related to the contact, and update the UI appropriately if the new contact address is authorized from another device.

4.4 Server-side processing of inbound moved notification

It is not required for Romeo's server to support this specification. However if Romeo's server does understand this extension, it SHOULD handle the inbound subscription request on behalf of Romeo's clients. This improves the user experience for Romeo, especially when he has multiple devices.
Broadly the server should follow exactly the same process as a client would. Specifically:

1. Receive subscription request with 'moved' payload.
2. Verify that the old JID has an approved subscription to the user (i.e. a subscription of 'both' or 'from').
3. Request moved statement from the old account JID.
4. Verify that the new JID in the moved statement matches the 'from' of the subscription request.

Listing 5: Romeo's server requests a moved statement from Juliet's old account

```
<iq type='get'
    from='romeo@capulet.example'
    to='juliet@im.example.net'
    id='83hKgF'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='urn:xmpp:moved:1'>
      <item id='current'/>
    </items>
  </pubsub>
</iq>
```

Listing 6: Juliet's old server returns the published moved statement to Romeo's server

```
<iq type='result'
    from='juliet@im.example.net'
```

```
    to='romeo@capulet.example/laptop'
    id='83hKgF'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='urn:xmpp:moved:1'>
      <item id='current'>
        <moved xmlns='urn:xmpp:moved:1'>
          <new-jid>juliet@capulet.example</new-jid>
        </moved>
      </item>
    </items>
  </pubsub>
</iq>
```

If verification fails (e.g. due to a missing or incorrect moved statement), the server MUST ignore the <moved/> element in the subscription request, and process the stanza as a normal subscription request. It MUST NOT strip the <moved/> element before forwarding, even if verification fails.
Upon successful verification, the server MUST NOT forward the stanza to Romeo's clients, but instead MUST create a roster entry for the new JID with a subscription of 'from' (sending out the appropriate roster push), and then auto-reply to the subscription request with a presence of type 'subscribed'.

Listing 7: Romeo's server notifies Romeo about the new roster entry

```
<iq id='a78b4q6ha463'
    to='romeo@montague.example/orchard'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='juliet@capulet.example'
          subscription='from'/>
  </query>
</iq>
```

After authorizing the new JID, the server should revoke the presence subscription of the old account.

Listing 8: Romeo's server notifies Juliet's old account about unsubscription

```
<presence to='juliet@im.example.net' type='unsubscribed'/>
```

Listing 9: Romeo's server notifies Romeo's clients about the old roster entry

```
<iq id='a78b4q6ha463'
    to='romeo@montague.example/orchard'
    type='set'>
  <query xmlns='jabber:iq:roster'>
    <item jid='juliet@im.example.net'
          subscription='to'/>
```

```
  </query>
</iq>
```

Finally, if the old JID was in the user's roster with a subscription of 'both', the server MUST also send a presence of type 'subscribe' to the new JID in order to seek establishment of a mutual subscription.

Listing 10: Romeo's server sends subscription request to Juliet's new JID

```
<presence type='subscribe' id='uc51xs63' from='romeo@montague.example'
    to='juliet@capulet.example'/>
```

5 Implementation Notes

5.1 Lifetime of moved statement

The moved statement is required for contacts to automatically verify the authenticity of moved notifications. After publishing a moved statement, the user should keep the statement published and the account active for long enough for contacts to switch to the user's new account.
It is not necessary to remain connected to the old account during the transition period. However the account must not be deleted, and the server must be available.
In the event that the move statement is unpublished, the account is deleted, or the server becomes unavailable, any contacts that have not yet transitioned to the user's new JID will be unable to verify the migration. Those contacts will have to manually approve the subscription from the user's new address.
Migration progress of contacts is obervable through subscription revocations to the old account, and subscription approvals to the new account.

5.2 Alternative verification methods

Future revisions of this document, or alternative documents, may define alternative verification methods. Such methods SHOULD be communicated via child elements of <moved/> in an appropriate namespace. As is usual throughout XMPP, entities MUST ignore unknown child elements of <moved/> in unrecognised namespaces.

6 Security Considerations

6.1 User considerations

The following are considerations for the user (exemplified by Juliet in this document):

- A malicious client or other entity with access to the user's account can perform a migration, potentially against the user's will and/or without their knowledge. Although this is a concern, any malicious actor with access to a user's account can abuse that access in many ways. Servers that support granting restricted access to accounts should consider blocking attempts to publish to the 'urn:xmpp:moved:1' PEP node from restricted entities.
- To avoid leaking the user's new JID to non-contacts, the PEP node containing the moved statement SHOULD be configured to use the "presence" access model (this is the default access model defined by PEP).

6.2 Contact considerations

The following are considerations for the contact (exemplified by Romeo in this document), and the contact's server:

- A presence subscription with a <moved/> is trivial for a malicious third-party to spoof. The verification methods discussed in this document MUST be followed to prevent accepting rogue subscription requests.
- It is important to verify that the original JID of the migrating user was already authorized to view presence before processing a migration.
- After successfully processing a migration, the original account should have its presence subscription revoked. This ensures that each JID may only be migrated once. Without this precaution the migration mechanism can be abused to introduce unlimited arbitrary JIDs to contacts' rosters. This precaution also ensures, if the old account ends up owned by a new entity, that they will not unexpectedly inherit a presence subscription.

7 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [2].

8 XMPP Registrar Considerations

This specification defines the following XML namespace:

- urn:xmpp:moved:1

[2] 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/>.

9 Design Considerations

9.1 Verification methods

There are two general approaches for verification - network-based verification, or cryptographic verification. Network-based verification (as described in this document) requests a verification statement from the user's old account. Cryptographic verification would check a move notification against a previously-established cryptographic identify of the user.
Network-based verification:

- Pro: Simple and easy to implement
- Con: depends on the original server being available and supporting PEP

Cryptographic verification:

- Pro: Can work even if the original server goes offline or begins blocking migration attempts.
- Con: More complex implementation.
- Con: Requires user and contacts to manage/track cryptographic keys and identies. It may be possible to piggyback on top of an existing cryptographic layer, e.g. OMEMO. However this would eliminate the server-side assistance, and OMEMO support is not universal among clients.

Ultimately this document defines a network-based verification method, but leaves an obvious path to extend the protocol with alternative verification methods (either in an update to this document, or defined by other documents).

10 XML Schema

To be done upon advancement to Draft.

11 Acknowledgements

This document was formerly driven by Tory Patnoe with the support and feedback provided by Doug Abbink, Mikhail Belov, Peter Saint-Andre, and Peter Sheu.
It has since been taken over by the current author who thanks Kim Alvefur, Maxime Buquet and Jonas Schäfer for their input and feedback on this specification.