# XEP-0291: Service Delegation

Justin Karneges
mailto:justin@affinix.com
xmpp:justin@andbit.net

2011-01-26
Version 0.1

| Status | Type | Short Name |
| --- | --- | --- |
| Deferred | Standards Track | NOT_YET_ASSIGNED |

This specification defines an approach for users to delegate certain services (e.g. pubsub) to alternative JIDs.

# Legal

## Copyright



## Permissions



## Warranty



## Liability



## Conformance

## Contents

# 1 Introduction

It is common to use XMPP user accounts for identity. Many features for user accounts have been proposed (for example, User Avatar (XEP-0084) [1], Microblogging Over XMPP (XEP-0277) [2], etc), and this list only grows as XMPP heads into social networking territory. However, not every XMPP server in the world supports every feature, nor can this ever be expected, and this limits the usefulness of the XMPP user account identity. Also, even when servers do support a feature it may not be clear how to bind an identity to it (see the never-solved problem of learning where a user's generic Publish-Subscribe (XEP-0060) [3] service is, in the absence of Personal Eventing Protocol (XEP-0163) [4]). Organizations such as Buddycloud and Livefyre have recognized the need to be able to offer features to existing standard XMPP accounts by offering proprietary service delegation mechanisms. This specification proposes a Standards Track solution for discovering external services associated with XMPP user identities.

# 2 How It Works

## 2.1 Discovering Delegate Services

To learn of delegate services for a user, query the bare JID of the user:

Listing 1: Query for delegate services

```
<iq type="get" from="alice@example.com/home" to="bob@example.com" id="
    d1">
  <query xmlns="urn:xmpp:tmp:delegate"/>
</iq>
```

Listing 2: Success response

```
<iq type="result" from="bob@example.com" to="alice@example.com/home"
    id="d1">
  <query xmlns="urn:xmpp:tmp:delegate">
    <service type="pubsub" jid="pubsub.example.net"/>
    <service type="chess" jid="bob@chess.example.net"/>
  </query>
</iq>
```

In the above example, we learn that "chess" related communication should go through the "bob@chess.example.net" JID rather than the user's own JID ("bob@example.com").

[1] XEP-0084: User Avatar <https://xmpp.org/extensions/xep-0084.html>.
[2] XEP-0277: Microblogging over XMPP <https://xmpp.org/extensions/xep-0277.html>.
[3] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[4] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.

## 2.2 Discovering Through a Third-Party Registry

As it is expected that most servers will not even support the ability to query user accounts for delegate services, it should be possible to contact a third-party or global registry holding the same information, to be queried in much the same way:

Listing 3: Query registry for delegate services

```
<iq type="get" from="alice@example.com/home" to="registry.example.com"
    id="r1">
  <query xmlns="urn:xmpp:tmp:delegate" jid="bob@example.com"/>
</iq>
```

Listing 4: Success response

```
<iq type="result" from="registry.example.com" to="alice@example.com/
   home" id="r1">
  <query xmlns="urn:xmpp:tmp:delegate">
    <service type="pubsub" jid="pubsub.example.net"/>
    <service type="chess" jid="bob@chess.example.net"/>
  </query>
</iq>
```

## 2.3 Managing Third-Party Registry Information

Here's how a user adds a mapping to the registry:

Listing 5: Adding an entry

```
<iq type="set" from="bob@example.com/home" to="registry.example.com"
   id="r2">
  <query xmlns="urn:xmpp:tmp:delegate">
    <service type="chess" jid="bob@chess.example.net"/>
  </query>
</iq>
```

Listing 6: Success response

```
<iq type="result" from="registry.example.com" to="bob@example.com/home
   " id="r2"/>
```

The registry will add the new service to the list of mapped services for the sender JID.
To remove a service from the list, submit without a jid attribute:

Listing 7: Removing an entry

```
<iq type="set" from="bob@example.com/home" to="registry.example.com"
   id="r3">
```

```
  <query xmlns="urn:xmpp:tmp:delegate">
    <service type="chess"/>
  </query>
</iq>
```

Listing 8: Success response

```
<iq type="result" from="registry.example.com" to="bob@example.com/home
    " id="r3"/>
```

### 2.4 Confirming With the Delegate

Whether a delegate JID is discovered directly or via registry, the mapping is an assertion made by the user only and not by the delegate. This may have security implications. For example, a third party should not allow the user to pose as the delegate, nor should the mapping be considered an endorsement by the delegate, since anyone can assert any delegate. However, these types of things could be allowed if third parties confirm with the delegate that the association exists.
Here's how to query a delegate JID to confirm if it is indeed associated with a specific user for a specific service type:

Listing 9: Confirming a delegation

```
<iq type="get" from="alice@example.com/home" to="bob@chess.example.net
    " id="c1">
  <check xmlns="urn:xmpp:tmp:delegate" type="chess" jid="bob@example.
      com"/>
</iq>
```

Note that the user JID expected to be associated with the delegate must be provided in the request. It is not possible to query a delegate to determine the user JID, since a single delegate may act for many users.
Success response means the association exists:

Listing 10: Success response

```
<iq type="result" from="bob@chess.example.net" to="alice@example.com/
    home" id="c1"/>
```

## 3 Implementation Notes

### 3.1 Use of Both Direct and Third-Party Queries

Applications that are configured to use a third-party registry SHOULD still be able to query user accounts directly. For performance reasons, it is recommended to query both the user

account and the registry simultaneously, and take whichever answer arrives first. If one of the queries results in an error, then the application should wait for the other query to complete before assuming no such delegate records exists.
If both queries return errors, or a success result does not contain an entry for some desired delegate service, it can be assumed that the desired service is provided by the user account itself (not delegated).

### 3.2 Caching

It is RECOMMENDED that discovery or confirmation of delegate information be cached indefinitely and refreshed no more frequently than every 24 hours. Data refreshing should not block access to existing information. If over 24 hours pass since the last time delegate information was needed, the application should continue to use the old data while independently firing off a task to refresh the data. This way, latency associated with a particular user delegation only occurs the first time a user is ever seen.

## 4 Security Considerations

As discussed above, a delegate mapping is an assertion by the user only. If the user should be allowed to act for the delegate, or if the user should be considered endorsed by the delegate, then the delegation needs to be confirmed first.

## 5 IANA Considerations

No interaction with the Internet Assigned Numbers Authority (IANA) [5] is required as a result of this document.

## 6 XMPP Registrar Considerations

### 6.1 Protocol Namespaces

This specification defines the following XML namespace:

- urn:xmpp:tmp:delegate

[5] 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/>.

Upon advancement of this specification from a status of Experimental to a status of Draft, the XMPP Registrar [6] shall add the foregoing namespace to the registry located at <https://xmpp.org/registrar/namespaces.html>, as described in Section 4 of XMPP Registrar Function (XEP-0053) [7].

### 6.2 Namespace Versioning

If the protocol defined in this specification undergoes a revision that is not fully backwards-compatible with an older version, the XMPP Registrar shall increment the protocol version number found at the end of the XML namespaces defined herein, as described in Section 4 of XEP-0053.

[6] 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/>.

[7] XEP-0053: XMPP Registrar Function <https://xmpp.org/extensions/xep-0053.html>.