XEP-0465: Pubsub Public Subscriptions

Jérôme Poisson
mailto:goffi@goffi.org
xmpp:goffi@jabber.fr

2022-07-25
Version 0.1.1

Status	Type	Short Name
Experimental	Standards Track	pps

This specification provides a way to make subscriptions to a node public

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Publish-Subscribe (XEP-0060) [1] as its name states has a mechanism to subscribe to a node. Only the owner of the node can retrieve the list of subscribers
It may be interesting for users to share publicly the nodes they have subscribed to, or who is subscribed to theirs: it's a quick way to discover center of interest of a user, or to discover new accounts/nodes related to a specific center of interest. This kind of feature is common in modern social networks and often named "following" and "followers". This XEP proposes a solution to implement this feature in XMPP while respecting privacy of users.
There is currently a XEP partially covering this problem with Pubsub Subscription (XEP-0330) [2]. This XEP has the advantage to be usable with a generic Pubsub service, but it has 2 flaws:

- it's only covering half of the problem: we get the pubsub nodes to which an entity is subscribed, but we don't know who is subscribed to a node
- clients need to keep the 'urn:xmpp:pubsub:subscription' node synchronized with subscriptions: if a subscription is removed from a node, it may stay present by mistake in 'urn:xmpp:pubsub:subscription' node.

This XEP fixes both issues.

2 Requirements

The design goal of this XEP are:

- let a user discover to which node an other user is publicly subscribed
- let a user discover who is publicly subscribed to a node
- take care of privacy: a user must declare a subscription public on purpose
- keep public subscription synchronized with nodes subscriptions

This XEP uses Pubsub Account Management (XEP-0376) [3] as only way to subscribe to a node and unsubscribe from a node, as it is necessary to keep track of subscriptions.

[1] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[2] XEP-0330: Pubsub Subscription <https://xmpp.org/extensions/xep-0330.html>.
[3] XEP-0376: Pubsub Account Management <https://xmpp.org/extensions/xep-0376.html>.

3 Glossary

In this documentation, **PAM service** refers to a PEP service implementing Pubsub Account Management (XEP-0376) [4].

4 Use Cases

4.1 Public Subscription

Romeo wants to subscribe to the blog of his cousin Benvolio and he wants to make it public, so other peoples can discover Benvolio blog more easily.
He does that as usual by sending a subscription request as explained in XEP-0376 §Subscribing but he adds a <public> element with the 'urn:xmpp:pps:0' namespace:

Listing 1: Romeo Makes a Public Subscription to Benvolio Blog

```
<iq type='set' id='sub1'>
  <pam xmln='urn:xmpp:pam:0' jid='benvolio@montague.lit'>
    <subscribe xmlns='http://jabber.org/protocol/pubsub'
        node='urn:xmpp:microblog:0'
        jid='romeo@montague.lit'>
        <public xmlns='urn:xmpp:pps:0'/>
    </subscribe>
  </pam>
</iq>
```

Romeo also wants to follow the blog of this girl that he met at the ball, however, he doesn't want yet to make it public for political reasons. He then does the subscription as usual and **does not** include the <public> element:

Listing 2: Romeo Makes a Non Public Subscription to Juliet Blog

```
<iq type='set' id='sub2'>
  <pam xmln='urn:xmpp:pam:0' jid='juliet@capulet.lit'>
    <subscribe xmlns='http://jabber.org/protocol/pubsub'
        node='urn:xmpp:microblog:0'
        jid='romeo@montague.lit'/>
  </pam>
</iq>
```

4.2 Retrieving Public Subscriptions

Mercutio is a friend of Romeo and he wants to know which nodes Romeo is subscribed to, as it may be a way to discover new and interesting peoples. To do this, Romeo's PAM service

[4] XEP-0376: Pubsub Account Management <https://xmpp.org/extensions/xep-0376.html>.

manages a special node named 'urn:xmpp:pps:subscriptions:0'. This node is created and managed by the PAM service itself, it can be subscribed to and unsubscribed from as an usual PubSub node, and it contains an item for each public subscription that has been made by node owner (Romeo in our example). Each items payload is a <subscription> element with the 'urn:xmpp:pps:0' namespace containing a 'node' attribute with the name of the subscribed node, and a 'jid' attribute with the JID of the pubsub service containing the subscribed node. The node owner can't add or retract items directly on the node: if Romeo wants to add or public subscription, it does this by doing a public subscription as explained in Public Subscription, and if he wants to retract a public subscription, he can do as explained in the next section. Result Set Management (XEP-0059) [5] and Pubsub Message Archive Management (XEP-0442) [6] apply normally if they are implemented.

Listing 3: Mercutio Get Public Subscriptions of Romeo

```
<iq type='get'
    from='mercutio@escalus.lit/play.456'
    to='romeo@montague.lit'
    id='get_pub_sub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='urn:xmpp:pps:subscriptions:0'/>
  </pubsub>
</iq>
```

Listing 4: Romeo's PAM Service Replies With Public Subscriptions

```
<iq type='result'
    from='romeo@montague.lit'
    to='mercutio@escalus.lit/play.456'
    id='get_pub_sub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='urn:xmpp:pps:subscriptions:0'>
      <item id='abcd'>
          <subscription xmlns='urn:xmpp:pps:0' node='
              urn:xmpp:microblog:0' jid='benvolio@montague.lit' />
      </item>
    </items>
  </pubsub>
</iq>
```

4.3 Retracting Public Subscriptions

Romeo can retract a public subscription in 2 ways:

[5] XEP-0059: Result Set Management <https://xmpp.org/extensions/xep-0059.html>.
[6] XEP-0442: Pubsub Message Archive Management <https://xmpp.org/extensions/xep-0442.html>.

- by unsubscribing entirely from the node as explained in XEP-0376 §Unsubscribing. In this case the PAM service remove the node from public subscriptions (it won't appear anymore if somebody retrieves subscriptions) and unsubscribe from the node.
- by subscribing again to the node **without** the <public> element, as explained in https://xmpp.org/extensions/xep-0376.html#subs. In this case the PAM service remove the node from public subscriptions, and forward the request to the pubsub service (so the pubsub service also knows that the subscription is not public anymore).

4.4 Retrieving Public Subscribers to a node

If Mercutio wants to know who is publicly subscribing to Romeo's blog, he request the PAM Service by using a special node managed by the service in a similar way as 'urn:xmpp:pps:0' node from Retrieving Public Subscriptions section (i.e. a node which can be subscribed to normally, but whose items can't be publised or retracted directly). This node is named by prefixing the name of the target node with 'urn:xmpp:pps:subscribers:0/'. So to check who is subscribed to Romeo's blog, Mercutio must request 'urn:xmpp:pps:subscribers:0/urn:xmpp:microblog:0' node. The service will answer with items whose payload is a <subscriber> element with the 'urn:xmpp:pps:0' namespace, and a 'jid' attribute whose value is the JID of the public subscriber:

Listing 5: Mercutio Get Public Subscribers of Romeo's blog

```
<iq type='get'
    from='mercutio@escalus.lit/play.456'
    to='romeo@montague.lit'
    id='get_pub_sub2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
      <items node='urn:xmpp:pps:subscribers:0/urn:xmpp:microblog:0'/>
  </pubsub>
</iq>
```

Listing 6: Romeo's PAM Service Replies With Public Subscribers

```
<iq type='result'
    from='romeo@montague.lit'
    to='mercutio@escalus.lit/play.456'
    id='get_pub_sub2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='{}'urn:xmpp:pps:subscribers:0/urn:xmpp:microblog:0>
      <item id='defg'>
          <subscriber xmlns='urn:xmpp:pps:0' jid='benvolio@montague.
              lit' />
      </item>
    </items>
  </pubsub>
</iq>
```

note: Public subscribers is not restricted to PAM service, if a generic pubsub service implements this XEP, it MUST also returns the public subscribers when the special node is requested.

5 Business Rules

If a user wants to create, purge or delete a special node used in this XEP, or if they want to manually publish or retract items, the service MUST return a <forbidden/> error to the user.

6 discovering support

If a PEP or Pubsub service supports the "Pubsub Public Subscriptions" protocol, it must advertize it by including the "urn:xmpp:pps:0" discovery feature (see Protocol Namespaces regarding issuance of one or more permanent namespaces) in response to a Service Discovery (XEP-0030) [7] information request:

Listing 7: service discovery information request

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

Listing 8: service discovery information response

```
<iq from='example.com'
    id='disco1'
    to='example.org'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>…

    <feature var='urn:xmpp:pps:0'/>…

  </query>
</iq>
```

7 Security Considerations

Publishing publicly subscriptions of a user has pricacy implications: those public subscriptions may be used by someone to get a user interests or to know they network of contacts.

[7] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

It may be used by bad actors for many reasons like advertising, or it may even be life threating in some countries/situation as it may be used to known political opinion, religion, sexual orientation, etc. A client SHOULD make the subscription public only if there is no doubt that this is what the user wants, by using an opt-in system, and SHOULD display a well visible warning about the consequences of making a subscription public.
By having subscription public, an entity JID can be checked or harvested by doing a request on the public subscriptions node. A client SHOULD display a warning clearly indicating that making subscriptions public makes its JID discoverable.
For the same reason, a server SHOULD respond identically to a pubsub request to public subscriptions node if the user doesn't exist or if they exist but they don't have any public subscriptions.

8 IANA Considerations

TODO

9 XMPP Registrar Considerations

TODO

10 XML Schema

TODO