XEP-0173: Pubsub Subscription Storage

Magnus Henoch
mailto:henoch@dtek.chalmers.se
xmpp:legoscia@jabber.cd.chalmers.se

2006-02-09
Version 0.1

Status	Type	Short Name
Deferred	Historical	pubsubs

This document defines an XMPP protocol extension for storing subscriptions to Pubsub nodes.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2020 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 Protocol **1**

4 Use Cases **2**

4.1 Retrieving Existing Subscriptions 2

4.2 Updating Subscriptions 2

4.3 Identifying Incoming Events 3

5 Security Considerations **4**

6 IANA Considerations **4**

7 XMPP Registrar Considerations **4**

8 XML Schema **4**

1 Introduction

Publish-Subscribe (XEP-0060) [1] allows Jabber entities to subscribe to various kinds of information, but provides no way of remembering which nodes a user has subscribed to. Other protocols (e.g. User Geolocation (XEP-0080) [2], User Avatar (XEP-0084) [3]) allow information about a certain entity to be published to a Pubsub node. These protocols use Service Discovery (XEP-0030) [4] to allow other entities to find the pubsub node used by a certain entity, but provide no way of performing the opposite mapping, from pubsub node to information source. This document attempts to fill that void, using Private XML Storage (XEP-0049) [5] for storing information about subscriptions.

2 Requirements

This protocol enables Jabber clients to do the following:

- Remember which pubsub nodes the entity has subscribed to, and what kind of information is available at each node
- Update the mappings when subscriptions are added or removed
- Correlate incoming pubsub events to subscriptions

3 Protocol

The <subscriptions/> element qualified by the 'storage:pubsubs' namespace is the root element used in the jabber:iq:private transactions. It has zero or more <subscription/> child elements, each of which MUST possess the following attributes:

- *jid* -- The JID of the pubsub service used
- *node* -- The pubsub node at which the data is available
- *subscription* -- The current subscription state, one of "none", "pending", "unconfigured" and "subscribed"

Additionally, the <subscription/> element MAY possess these attributes:

[1] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[2] XEP-0080: User Geolocation <https://xmpp.org/extensions/xep-0080.html>.
[3] XEP-0084: User Avatar <https://xmpp.org/extensions/xep-0084.html>.
[4] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[5] XEP-0049: Private XML Storage <https://xmpp.org/extensions/xep-0049.html>.

- *resource* -- The resource that is subscribed to this pubsub node. If the 'resource' attribute is absent, the bare JID is subscribed.
- *user* -- The JID of the owner of this particular piece of data
- *targetns* -- The namespace of the data in question

4 Use Cases

4.1 Retrieving Existing Subscriptions

In this example, the user already has a subscription to Juliet's geolocation, possibly established through another client.

Listing 1: Client requests existing subscriptions

```
<iq type='get' id='retrieve1'>
  <query xmlns='jabber:iq:private'>
    <subscriptions xmlns='storage:pubsubs'/>
  </query>
</iq>
```

Listing 2: Server returns existing subscriptions

```
<iq type='result' id='retrieve1'>
  <query xmlns='jabber:iq:private'>
    <subscriptions xmlns='storage:pubsubs'>
      <subscription user='juliet@capulet.com'
                    jid='pubsub.capulet.com'
                    node='juliet/geoloc'
                    targetns='http://jabber.org/protocol/geoloc'
                    subscription='subscribed'/>
    </subscriptions>
  </query>
</iq>
```

4.2 Updating Subscriptions

Due to the nature of XEP-0049, incremental updates are not possible; a client MUST send the entire <subscriptions/> node for each update. Before performing the update, the client SHOULD retrieve the stored subscriptions, and incorporate any changes.
In this example, the user has just subscribed to Romeo's tune (see User Tune (XEP-0118) [6]). Assuming that retrieving happened as in the previous use case, updating the subscriptions proceeds as follows:

[6] XEP-0118: User Tune <https://xmpp.org/extensions/xep-0118.html>.

Listing 3: Client sends updated subscriptions

```
<iq type='set' id='update1'>
  <query xmlns='jabber:iq:private'>
    <subscriptions xmlns='storage:pubsubs'>
      <subscription user='juliet@capulet.com'
                    jid='pubsub.capulet.com'
                    node='juliet/geoloc'
                    targetns='http://jabber.org/protocol/geoloc'
                    subscription='subscribed'/>
      <subscription user='romeo@montague.net'
                    jid='pubsub.montague.net'
                    node='5017cdc9f4a3d1450445c9096064e459'
                    targetns='http://jabber.org/protocol/tune'
                    subscription='subscribed'/>
    </subscriptions>
  </query>
</iq>
```

Listing 4: Server reports success

```
<iq type='result' id='update1'/>
```

4.3 Identifying Incoming Events

Having recorded the retrieved mappings, the client is now prepared to identify incoming pubsub events. Assume that the following event arrives:

Listing 5: Client receives pubsub event

```
<message
    from='pubsub.montague.net'
    to='mercutio@shakespeare.lit'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='5017cdc9f4a3d1450445c9096064e459'>
      <item id='current'>
        <tune xmlns='http://jabber.org/protocol/tune'>
          <artist>Yes</artist>
          <title>Heart of the Sunrise</title>
          <source>Yessongs</source>
          <track>3</track>
          <length>686</length>
        </tune>
      </item>
    </items>
  </event>
</message>
```

The client now knows that this information comes from romeo@montague.net.

5 Security Considerations

Pubsub events offer an opportunity to spoof sender addresses e.g. through 'replyto' data (as specified by the Extended Stanza Addressing (XEP-0033) [7] protocol). This protocol attempts to close that hole. It does so by the following rules and assumptions:

- A client MUST add mappings (i.e. associations between a publisher's JID and a pubsub node) only from trustworthy sources, i.e. published disco items (see Service Discovery (XEP-0030) [8]). This relies on disco information not being cracked or falsified.
- A client MUST retrieve mappings only from trustworthy sources, i.e. private XML storage. This assumes that no-one but the user is able to change such information.

6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [9].

7 XMPP Registrar Considerations

No namespaces or parameters need to be registered with the XMPP Registrar [10] as a result of this document.

8 XML Schema

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

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

  <xs:element name='subscriptions'>
```

[7] XEP-0033: Extended Stanza Addressing <https://xmpp.org/extensions/xep-0033.html>.

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

[9] 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] 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/>.

```
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='subscription' minOccurs='0' maxOccurs='
            unbounded'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='subscription'>
    <xs:complexType>
      <xs:attribute name='jid' type='xs:string' use='required'/>
      <xs:attribute name='node' type='xs:string' use='required'/>
      <xs:attribute name='subscription' use='required'>
        <xs:simpleType>
          <xs:restriction base='xs:NCName'>
            <xs:enumeration value='none'/>
            <xs:enumeration value='pending'/>
            <xs:enumeration value='subscribed'/>
            <xs:enumeration value='unconfigured'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name='resource' type='xs:string' use='optional'/>
      <xs:attribute name='user' type='xs:string' use='optional'/>
      <xs:attribute name='targetns' type='xs:string' use='optional'/>
    </xs:complexType>
  </xs:element>

</xs:schema>
```