XEP-0223: Persistent Storage of Private Data via PubSub

Peter Saint-Andre
mailto:xsf@stpeter.im
xmpp:peter@jabber.org
http://stpeter.im/

2008-09-08
Version 1.0

Status	Type	Short Name
Active	Informational	N/A

This specification defines best practices for using the XMPP publish-subscribe extension to persistently store private information such as bookmarks and client configuration options.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

1.1 Motivation

Personal Eventing Protocol (XEP-0163) [1] introduced the idea of a virtual Publish-Subscribe (XEP-0060) [2] service associated with an IM user's bare JID <localpart@domain.tld>. However, the default configuration of PEP nodes is not optimized for the persistent storage of data objects that are meant to be accessed only by the account owner, à la Private XML Storage (XEP-0049) [3]. Therefore this document defines a set of best practices that enable IM users to persistently store private information at their virtual pubsub service; in effect, we "subclass" PEP by showing how a particular pubsub node can be configured for storing private data.

1.2 How It Works

Imagine that you are a Shakespearean character named Juliet and that you want to persistently store some private information such as bookmarks (Bookmark Storage (XEP-0048) [4]).
We assume that your server (capulet.lit) supports PEP along with the "publish-options" feature, and that your client discovered that support when you logged in.
You want to start bookmarking Multi-User Chat (XEP-0045) [5] rooms, so your client stores that data privately.

Listing 1: Storing bookmarks

```
<iq from='juliet@capulet.lit/balcony' type='set' id='pdp1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='storage:bookmarks'>
      <item>
        <conference xmlns='storage:bookmarks'
                    autojoin='true'
                    jid='theplay@conference.shakespeare.lit'
                    name='The Play&apos;s the Thing'>
          <nick>JC</nick>
          <password>Gl0b3</password>
        </conference>
      </item>
    </publish>
    <publish-options>
      <x xmlns='jabber:x:data' type='submit'>
        <field var='FORM_TYPE' type='hidden'>
          <value>http://jabber.org/protocol/pubsub#publish-options</
              value>
```

[1] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.
[2] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[3] XEP-0049: Private XML Storage <https://xmpp.org/extensions/xep-0049.html>.
[4] XEP-0048: Bookmark Storage <https://xmpp.org/extensions/xep-0048.html>.
[5] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.

```
        </field>
        <field var='pubsub#persist_items'>
          <value>true</value>
        </field>
        <field var='pubsub#access_model'>
          <value>whitelist</value>
        </field>
      </x>
    </publish-options>
  </pubsub>
</iq>
```

Your publish request is a standard pubsub request except that:

1. The item is persisted (pubsub#persist_items is set to true).
2. In this case, access is limited to yourself (the "whitelist" access model defaults to allowing access for the account owner, i.e., you).

If all goes well (see Storing Data), your bookmarks will be stored and the information will be pushed out to all of your resources (here "balcony" and "chamber").

Listing 2: Publisher receives event notification

```
<message from='juliet@capulet.lit'
         to='juliet@capulet.lit/balcony'
         type='headline'
         id='bmfoo1'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='storage:bookmarks'>
      <item id='SomeID'>
        <conference xmlns='storage:bookmarks'
                    autojoin='true'
                    jid='theplay@conference.shakespeare.lit'
                    name='The Play&apos;s the Thing'>
          <nick>JC</nick>
          <password>Gl0b3</password>
        </conference>
      </item>
    </items>
  </event>
</message>

<message from='juliet@capulet.lit'
         to='juliet@capulet.lit/chamber'
         type='headline'
         id='bmfoo2'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
```

```
    <items node='storage:bookmarks'>
      <item id='SomeID'>
        <conference xmlns='storage:bookmarks'
                    autojoin='true'
                    jid='theplay@conference.shakespeare.lit'
                    name='The Play&apos;s the Thing'>
          <nick>JC</nick>
          <password>Gl0b3</password>
        </conference>
      </item>
    </items>
  </event>
</message>
```

So that's the general idea.

2 Concepts and Approach

The best practices defined herein re-use the concepts already defined in XEP-0060 and XEP-0163. In order to optimize for object persistence of private information instead of transient event notifications related to semi-public data, a node MUST be configured as follows:

1. Set pubsub#persist_items to true.
2. Set pubsub#access_model to "whitelist".

3 Storing Data

An account owner publishes an item to a node by following the protocol specified in XEP-0060:

Listing 3: Account owner stores data

```
<iq from='juliet@capulet.lit/balcony' type='set' id='pdp1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='storage:bookmarks'>
      <item>
        <conference xmlns='storage:bookmarks'
                    autojoin='true'
                    jid='theplay@conference.shakespeare.lit'
                    name='The Play&apos;s the Thing'>
          <nick>JC</nick>
          <password>Gl0b3</password>
        </conference>
      </item>
    </publish>
```

```
  </pubsub>
</iq>
```

If the node does not already exist, the virtual pubsub service MUST create the node. As described in XEP-0163, this "auto-create" feature (defined in XEP-0060) MUST be supported by a PEP service. (Naturally, the account owner's client MAY follow the node creation use case specified in XEP-0060 before attempting to publish an item.)
In order for the client to reliably persist private information, the virtual pubsub service must also support the "publish-options" feature defined in XEP-0060. Typically, a client will publish with options so that the object is privately stored.

Listing 4: Storing bookmarks

```
<iq from='juliet@capulet.lit/balcony' type='set' id='pdp1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='storage:bookmarks'>
      <item>
        <conference xmlns='storage:bookmarks'
                    autojoin='true'
                    jid='theplay@conference.shakespeare.lit'
                    name='The Play&apos;s the Thing'>
          <nick>JC</nick>
          <password>Gl0b3</password>
        </conference>
      </item>
    </publish>
    <publish-options>
      <x xmlns='jabber:x:data' type='submit'>
        <field var='FORM_TYPE' type='hidden'>
          <value>http://jabber.org/protocol/pubsub#publish-options</
              value>
        </field>
        <field var='pubsub#persist_items'>
          <value>true</value>
        </field>
        <field var='pubsub#access_model'>
          <value>whitelist</value>
        </field>
      </x>
    </publish-options>
  </pubsub>
</iq>
```

If the publication logic dictates that event notifications shall be sent, the account owner's server generates notifications and sends them to all appropriate entities as described in the Receiving Event Notifications section of XEP-0163.

4 Composition

Each item published to the node is a logically separate instance of the data to be stored. It is the responsibility of the publishing and receiving entities to construct a complete view of all such items, if desired. For example, each bookmark published to a private data node is a separate piece of data, whereas the history of all items published to the node provides a complete list of the user's bookmarks. This history may include items that are republished with an existing ItemID (thus overwriting the previous version of that item).

5 Determining Support

Before an account owner attempts to complete any of the use cases defined herein, its client SHOULD verify that the account owner's server supports both PEP and the "publish-options" feature; to do so, it MUST send a Service Discovery (XEP-0030) [6] information request to the server (or cache Entity Capabilities information received from the server).

Listing 5: Account owner queries server regarding protocol support

```
<iq from='juliet@capulet.lit/balcony'
    to='capulet.lit'
    id='disco1'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

The server MUST return an identity of "pubsub/pep" and include the "publish-options" feauture in the list of the namespaces and other features it supports:

Listing 6: Server communicates protocol support

```
<iq from='capulet.lit'
    to='juliet@capulet.lit/balcony'
    id='disco1'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='server' type='im'/>
    <identity category='pubsub' type='pep'/>
    ...
    <feature var='http://jabber.org/protocol/pubsub#publish-options'/>
    ...
  </query>
</iq>
```

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

6 Security Considerations

This document introduces no security considerations above and beyond those specified in XEP-0060 and XEP-0163.

7 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [7].

8 XMPP Registrar Considerations

This document requires no interaction with the XMPP Registrar [8].

[7] 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/>.

[8] 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/>.