XEP-0254: PubSub Queueing

Joe Hildebrand
mailto:jhildebr@cisco.com
xmpp:hildjj@jabber.org

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

2008-11-13
Version 0.1

Status	Type	Short Name
Deferred	Standards Track	NOT_YET_ASSIGNED

This specification defines an extension to XMPP publish-subscribe for queueing information at a node.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction **1**

2 How It Works **1**

2.1 Subscribing 1

2.2 Assigning an Item 3

2.3 Deleting an Item from the Queue 4

2.4 Unlocking an Item 5

3 Determining Support **8**

4 Implementation Notes **8**

5 Security Considerations **8**

6 IANA Considerations **8**

7 XMPP Registrar Considerations **9**

7.1 Protocol Namespaces 9

7.2 Protocol Versioning 9

7.3 Service Discovery Features 9

1 Introduction

The Publish-Subscribe (XEP-0060) [1] extension to XMPP provides a comprehensive technology for alerts, notifications, data syndication, rich presence, and other real-time messaging use cases. In terms of traditional publish-subscribe systems like Java Message Service (JMS), the core XMPP PubSub specification covers the Observer design pattern only; however, traditional publish-subscribe systems often include support for a second design pattern, usually called the "point-to-point" or "queueing" pattern. [2] This specification defines a few small extensions to XMPP PubSub that enable support for a queueing mode in XMPP. The queueing mode is an add-on feature that a service can support, and that a node owner can enable if supported by the service. The feature name is "urn:xmpp:pubsub:queueing:0".

2 How It Works

2.1 Subscribing

If a node has enabled support for the queueing mode, in response to a subscription request without configuration options it MUST return an IQ-error containing a subscription options form; this form MUST include the "queue_requests" field (which specifies the number of parallel requests a subscriber is willing to process).

Listing 1: Service indicates that subscription configuration is required

```
<iq from='workflows.shakespeare.lit'
    id='sub1'
    to='workerbee237@shakespeare.lit/foo'
    type='error'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscribe
        jid='workerbee237@shakespeare.lit'
        node='a290fjsl29j19kjb'/>
    <options
        jid='workerbee237@shakespeare.lit'
        node='a290fjsl29j19kjb'>
      <x xmlns='jabber:x:data' type='form'>
        <field var='FORM_TYPE' type='hidden'>
          <value>http://jabber.org/protocol/pubsub#subscribe_options</
              value>
        </field>
        <field type='text-single' var='pubsub#queue_requests'>
          <required/>
        </field>
      </x>
```

[1] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[2] See for instance <http://en.wikipedia.org/wiki/Java_Message_Service>.

```
    </options>
  </pubsub>
  <error type='modify'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <configuration-required xmlns='http://jabber.org/protocol/pubsub#
        errors'/>
  </error>
</iq>
```

The subscriber would then send a new subscription request, this time with options.

Listing 2: Subscribing with options

```
<iq from='workerbee237@shakespeare.lit/foo'
    id='sub2'
    to='workflows.shakespeare.lit'
    type='set'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscribe
        jid='workerbee237@shakespeare.lit'
        node='a290fjsl29j19kjb'/>
    <options
        jid='workerbee237@shakespeare.lit'
        node='a290fjsl29j19kjb'>
      <x xmlns='jabber:x:data' type='submit'>
        <field var='FORM_TYPE' type='hidden'>
          <value>http://jabber.org/protocol/pubsub#subscribe_options</
              value>
        </field>
        <field var='pubsub#queue_requests'>
          <value>5</value>
        </field>
      </x>
    </options>
  </pubsub>
</iq>
```

If the subscription request can be processed successfully, the service returns an IQ-result and includes the configuration options established during the negotiation.

Listing 3: Service replies with success (including configuration options)

```
<iq from='workflows.shakespeare.lit'
    id='sub2'
    to='workerbee237@shakespeare.lit/foo'
    type='result'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscription
        jid='workerbee237@shakespeare.lit'
```

```
        node='a290fjsl29j19kjb'
        subid='ba49252aaa4f5d320c24d3766f0bdcade78c78d3'
        subscription='subscribed'/>
    <options>
      <x xmlns='jabber:x:data' type='result'>
        <field var='FORM_TYPE' type='hidden'>
          <value>http://jabber.org/protocol/pubsub#subscribe_options</
              value>
        </field>
        <field var='pubsub#queue_requests'>
          <value>5</value>
        </field>
      </x>
    </options>
  </pubsub>
</iq>
```

2.2 Assigning an Item

At any time, a publisher can push an item to the queue node.

Listing 4: Publisher publishes an item

```
<iq from='engine.shakespeare.lit'
    id='pub1'
    to='workflow.shakespeare.lit'
    type='set'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='a290fjsl29j19kjb'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'>
        <example xmlns='urn:xmpp:example'>payload</example>
      </item>
    </publish>
  </pubsub>
</iq>
```

The item is published to *one* of the subscribers.

Listing 5: One subscriber receives a notification

```
<message from='workflow.shakespeare.lit'
         id='foo'
         to='workerbee237@shakespeare.lit'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='a290fjsl29j19kjb'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'>
        <example xmlns='urn:xmpp:example'>payload</example>
      </item>
```

```
    </items>
  </event>
</message>
```

2.3 Deleting an Item from the Queue

When the subscriber that received the item has successfully processed it (whatever that means in the context of the queue), the subscriber deletes the item from the queue.

Listing 6: Entity deletes an item from a node

```
<iq from='workerbee237@shakespeare.lit/foo'
    id='delete1'
    to='workflows.shakespeare.lit'
    type='get'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <retract node='a290fjsl29j19kjb'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'/>
    </retract>
  </pubsub>
</iq>
```

In the context of a queue node, the service MUST treat a delete request from a subscriber that received the item as if the sender were a publisher; i.e., it MUST delete the item from the queue and notify only this subscriber that the item has been deleted.

Listing 7: Subscriber receives delete notification

```
<message from='workflow.shakespeare.lit'
         id='bar'
         to='workerbee237@shakespeare.lit'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='a290fjsl29j19kjb'>
      <retract id='ae890ac52d0df67ed7cfdf51b644e901'/>
    </items>
  </event>
</message>
```

Note: The subscriber SHOULD NOT commit any pending transactions until it receives the delete notification.
If the item does not exist, the service MUST return an <item-not-found/> error as described in XEP-0060.
If the entity that attempts to delete the item is not the subscriber that received the item, the service MUST return a <forbidden/> error as described in XEP-0060.
If the item is locked by another subscriber, the service MUST return a <conflict/> error (this flow is not defined in XEP-0060.

Listing 8: Item is locked by another subscriber

```
<iq from='workflows.shakespeare.lit'
    id='delete1'
    to='workerbee237@shakespeare.lit/foo'
    type='get'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <retract node='a290fjsl29j19kjb'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'/>
    </retract>
  </pubsub>
  <error type='auth'>
    <forbidden xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the subscriber that received the item attempts to delete the item but the item is no longer locked by the subscriber (e.g., because of a race condition or a lost notification), the service MUST return an <unexpected-request/> error (this flow is not defined in XEP-0060.

Listing 9: Item is no longer locked by subscriber

```
<iq from='workflows.shakespeare.lit'
    id='delete1'
    to='workerbee237@shakespeare.lit/foo'
    type='get'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <retract node='a290fjsl29j19kjb'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'/>
    </retract>
  </pubsub>
  <error type='wait'>
    <unexpected-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

2.4 Unlocking an Item

The subscriber might determine that it cannot process the item (whatever that means in the context of the queue); if so, the subscriber unlocks the item.

Listing 10: Entity unlocks an item

```
<iq from='workerbee237@shakespeare.lit/foo'
    id='unlock1'
    to='workflows.shakespeare.lit'
    type='get'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
```

```
    <unlock xmlns='urn:xmpp:pubsub:queueing:0'
            node='a290fjsl29j19kjb'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'/>
    </unlock>
  </pubsub>
</iq>
```

The service then MUST unlock the item and notify only this subscriber that the item has been unlocked.

Listing 11: Subscriber receives unlock notification

```
<message from='workflow.shakespeare.lit'
         id='baz'
         to='workerbee237@shakespeare.lit'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='a290fjsl29j19kjb'>
      <unlock xmlns='urn:xmpp:queueing:0'
              id='ae890ac52d0df67ed7cfdf51b644e901'/>
    </items>
  </event>
</message>
```

When an item is unlocked, the service would then send a publish notification to another subscriber according to application-specific logic for determining the ”next” subscriber.
If the item does not exist, the service MUST return an <item-not-found/> error.

Listing 12: Item does not exist

```
<iq from='workflows.shakespeare.lit'
    id='delete1'
    to='workerbee237@shakespeare.lit/foo'
    type='error'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <unlock xmlns='urn:xmpp:pubsub:queueing:0'
            node='a290fjsl29j19kjb'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'/>
    </unlock>
  </pubsub>
  <error type='cancel'>
    <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the entity that attempts to unlock the item is not the subscriber that received the item, the service MUST return a <forbidden/> error.

Listing 13: Requesting entity did not receive item

```
<iq from='workflows.shakespeare.lit'
    id='delete1'
    to='workerbee237@shakespeare.lit/foo'
    type='error'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <unlock xmlns='urn:xmpp:pubsub:queueing:0'
            node='a290fjsl29j19kjb'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'/>
    </unlock>
  </pubsub>
  <error type='auth'>
    <forbidden xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the item is locked by another subscriber, the service MUST return a <conflict/> error.

Listing 14: Item is locked by another subscriber

```
<iq from='workflows.shakespeare.lit'
    id='delete1'
    to='workerbee237@shakespeare.lit/foo'
    type='error'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <unlock xmlns='urn:xmpp:pubsub:queueing:0'
            node='a290fjsl29j19kjb'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'/>
    </unlock>
  </pubsub>
  <error type='cancel'>
    <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

If the subscriber that received the item attempts to unlock the item but the item is no longer locked by the subscriber (e.g., because of a race condition or a lost notification), the service MUST return an <unexpected-request/> error.

Listing 15: Item is no longer locked by subscriber

```
<iq from='workflows.shakespeare.lit'
    id='delete1'
    to='workerbee237@shakespeare.lit/foo'
    type='error'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <unlock xmlns='urn:xmpp:pubsub:queueing:0'
            node='a290fjsl29j19kjb'>
      <item id='ae890ac52d0df67ed7cfdf51b644e901'/>
```

```
    </unlock>
  </pubsub>
  <error type='wait'>
    <unexpected-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

3 Determining Support

If a pubsub service supports the queueing mode, it MUST advertise support for the "urn:xmpp:pubsub:queueing:0" namespace in response to Service Discovery (XEP-0030) [3] information requests.

4 Implementation Notes

If the service receives unavailable presence from a subscriber, it SHOULD unlock all outstanding queue items associated with the subscriber and unsubscribe the subscriber to prevent delivery of further publish notifications.
If a subscriber cannot process queue items because of an unrecoverable error (e.g., disk full), the subscriber SHOULD unsubscribe and then unlock all of its outstanding queue items.
If the service does not receive a delete or unlock request from a subscriber that received a queue item in a configurable amount of time, it SHOULD timeout the request, send an unlock notification to the subscriber, and send a publish notification to the "next" subscriber.

5 Security Considerations

To follow.

6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [4].

[3] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[4] 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/>.

7 XMPP Registrar Considerations

7.1 Protocol Namespaces

This specification defines the following XML namespace:

- urn:xmpp:pubsub:queueing:0

Upon advancement of this specification from a status of Experimental to a status of Draft, the XMPP Registrar [5] shall add the foregoing namespaces to the registry located at <https://xmpp.org/registrar/namespaces.html>, as described in Section 4 of XMPP Registrar Function (XEP-0053) [6].

7.2 Protocol 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.

7.3 Service Discovery Features

The XMPP Registrar [7] maintains a registry of service discovery features (see <https://xmpp.org/registrar/disco-features.html>), which includes a number of features that can be returned by pubsub services. The following registry submission supplements the existing list.

```
<var>
  <name>urn:xmpp:pubsub:queueing:0</name>
  <desc>The node or service supports the queueing mode.</desc>
  <doc>XEP-xxxx</doc>
</var>
```

[5] 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/>.

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

[7] 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/>.