Network Working Group	P. Saint-Andre	
Internet-Draft	XMPP Standards Foundation	
Intended status: Informational	J. Hildebrand	
Expires: November 9, 2008	Jabber, Inc.	
	B. Wyman	
	May 08, 2008	

TOC

Atomsub: Transporting Atom Notifications over the Publish-Subscribe Extension to the Extensible Messaging and Presence Protocol (XMPP)
draft-saintandre-atompub-notify-07

Status of this Memo

Abstract

This memo describes a method for notifying interested parties about changes in syndicated information encapsulated in the Atom feed format, where such notifications are delivered via an extension to the Extensible Messaging and Presence Protocol (XMPP) for publish-subscribe functionality.

Table of Contents

---

1. Introduction

TOC

1.1. Overview

TOC

The Atom Publishing Format and Protocol Working Group has developed two technologies relevant to content syndication:

1. An XML data format for syndication of information about periodically-updated resources (such as weblog entries and news stories) available on the World Wide Web (see [ATOM-FORMAT] (Nottingham, M. and R. Sayre, “The Atom Syndication Format,” December 2005.)).

2. A protocol for publishing, editing, deleting, and otherwise managing such resources (see [ATOM-PROTOCOL] (Gregorio, J. and B. de hOra, “The Atom Publishing Protocol,” October 2007.)).

Content syndication follows a classic "observer" or "publish-subscribe" software design pattern: a person or application publishes information to a "channel", and an event notification (or the data itself) is broadcasted to all those who are interested in knowing when such information is published or modified. On the Internet today, publication of periodically-updated resources is handled by means of standard technologies such as [HTTP] (Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1,” June 1999.), and it is not envisioned that this will change since [ATOM-PROTOCOL] (Gregorio, J. and B. de hOra, “The Atom Publishing Protocol,” October 2007.) specifies the use of HTTP for publication. However, existing methods for learning that a resource has been updated are currently limited to "polling" for changes via HTTP, which is inherently inefficient. What is needed is a technology that can be relied on to "push" information only when a resource undergoes a state change, and only to those who are interested in learning about such state changes.
One possible technology for doing so is email, since [SMTP] (Klensin, J., “Simple Mail Transfer Protocol,” April 2001.) provides a way to initiate the sending of information from "publishers" to "subscribers" (think, for example, of email lists such as those used to announce newly-published RFCs). While email is one possible solution, it is not

necessarily the best solution for Atom; in particular, [ATOM-FORMAT] (Nottingham, M. and R. Sayre, “The Atom Syndication Format,” December 2005.) defines an XML data format for content syndication, which implies that it might be beneficial to use a native XML delivery mechanism rather than to attach a special XML media type to email messages. Thankfully, a specialized XML delivery protocol has been developed through the IETF: the Extensible Messaging and Presence Protocol (XMPP) as specified in [XMPP-CORE] (Saint-Andre, P., “Extensible Messaging and Presence Protocol (XMPP): Core,” October 2004.). XMPP has the added benefit of being optimized for near-real-time data delivery, which may be important in applications of Atom that require subscribers to be notified about syndicated content in a highly timely manner.

While the semantics of a normal XMPP <message/> element may be suitable for Atom content notifications, there also exists an XMPP extension that provides more structured communications in the context of information "channels" or "nodes" of the kind that are used in typical content syndication technologies, where an interested entity can subscribe to that channel or node and thus receive notifications related to the topic of interest. This extension is specified in [XMPP-PUBSUB] (Millard, P., Saint-Andre, P., and R. Meijer, “Publish-Subscribe,” March 2008.) and may be especially useful for delivering notifications related to changes in Atom resources. Therefore, this memo describes a method for notifying interested parties about changes in syndicated information encapsulated in the Atom feed format, where such notifications are delivered via the XMPP publish-subscribe extension.

1.2. Terminology

TOC

This document inherits terminology from [ATOM-FORMAT] (Nottingham, M. and R. Sayre, “The Atom Syndication Format,” December 2005.), [XMPP-CORE] (Saint-Andre, P., “Extensible Messaging and Presence Protocol (XMPP): Core,” October 2004.), and [XMPP-PUBSUB] (Millard, P., Saint-Andre, P., and R. Meijer, “Publish-Subscribe,” March 2008.).

The capitalized key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.) [TERMS].

---

2. Process Flows

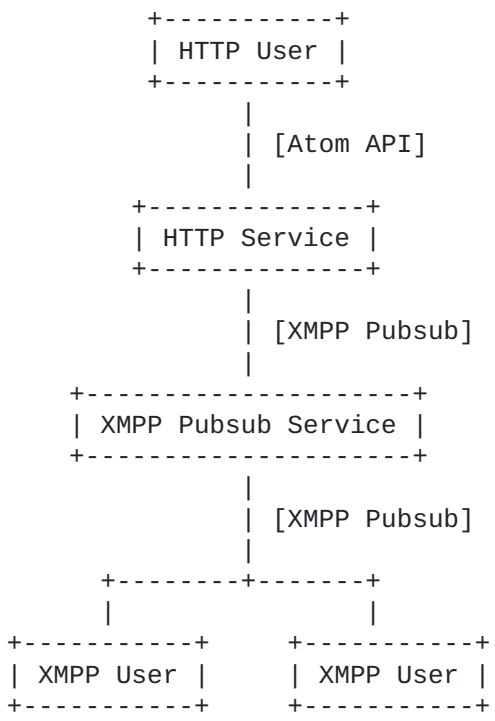
TOC

2.1. Overview

TOC

The following process flows demonstrate how Atom-formatted data (specifically, feed entries) can be delivered using the XMPP pubsub extension. The actors in these process flows are an application and one

or more XMPP users. The application acts as a translator between HTTP and XMPP, since it generates XMPP pubsub requests when certain events occur at an Atom-aware HTTP service (e.g., an HTTP POST to create a new dynamic resource). The XMPP pubsub service then translates those pubsub requests into notifications that are sent to a potentially large number of XMPP users who have subscribed to such events (e.g., who have asked to receive an XMPP notification whenever a new dynamic resource is created for a certain Atom "channel"). Of course, an XMPP user is not necessarily a human, and could represent another application on the XMPP network (e.g., a chatroom, a bot, or a content management system). Note well that an HTTP user (e.g., a weblog author) would still publish information using the methods defined in [ATOM-PROTOCOL] (Gregorio, J. and B. de hOra, "The Atom Publishing Protocol," October 2007.); the process flows described herein enable the HTTP service with which an HTTP user interacts to generate notifications that are delivered via an XMPP pubsub service to a potentially large number of XMPP users who want to receive such information.
We can visualize the architecture as follows:



In the examples shown below, we stipulate the following particulars:

*The XMPP address of the HTTP Service is "atompub.example.org".

*The XMPP address of the XMPP Pubsub Service is "pubsub.example.com".

*The NodeID of the XMPP pubsub node to which the HTTP Service publishes and to which the XMPP Users subscribe is "an-atom-node".

*The ItemID of the XMPP pubsub item published by the HTTP Service is "70b2a83be71dfca04df91133d953fb22b41b4267".

*The XMPP addresses of the XMPP Users who are subscribed to the node are "alice@example.net" and "bob@example.com".

2.2. Notification of Entry Creation

TOC

An implementation MUST support notifications related to creation of an entry.
When a content author publishes a new dynamic resource, many entities may be interested in learning that the resource is now available. The process flow is as follows:

*Author publishes a new entry to the HTTP service via the Atom API.

*The HTTP service sends data for the new Atom entry in an XMPP pubsub "publish" request to a specific node at the XMPP pubsub service. (Note: If the entry may be copied from one feed to another, e.g., in the generation of "synthetic" feeds, the entry SHOULD contain an atom:source element to ensure consistent metadata.)

*The XMPP pubsub service sends an XMPP message notification to each XMPP entity that is subscribed to the pubsub node.

The result is that the XMPP subscribers will receive something close to real-time notification whenever a new feed entry has been published. Obviously the first step is out of scope for this memo, since it is described in [ATOM-PROTOCOL] (Gregorio, J. and B. de hOra, “The Atom Publishing Protocol,” October 2007.). The XMPP protocols for the last two steps are shown below.
First the HTTP service sends an XMPP pubsub "publish" request to the XMPP pubsub service:

```
<iq type='set'
    from='atompub.example.org'
    to='pubsub.example.com'
    id='publish1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='an-atom-node'>
      <item id='70b2a83be71dfca04df91133d953fb22b41b4267'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <source>
            <title>Example Feed</title>
            <link href='http://example.org/'/>
            <link rel='self'
                  type='application/atom+xml'
                  href='http://example.org/atom.xml'/>
            <id>tag:example.org,2003:home</id>
            <updated>2003-12-13T18:30:02Z</updated>
            <author>
              <name>John Doe</name>
            </author>
          </source>
          <title>Atom-Powered Robots Run Amuck</title>
          <summary>Asimov's First Law horribly violated!</summary>
          <link rel='alternate'
                type='text/html'
                href='http://example.org/2003/12/13/atom03'/>
          <id>tag:example.org,2003:entry-32397</id>
          <published>2003-12-13T18:30:02Z</published>
          <updated>2003-12-13T18:30:02Z</updated>
        </entry>
      </item>
    </publish>
  </pubsub>
</iq>
```

The XMPP pubsub service then sends a pubsub notification to each XMPP subscriber; depending on pubsub node configuration, the notification may or may not contain the Atom payload (we assume here that the payload will be included).

```
<message from='pubsub.example.com'
         to='alice@example.net'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='an-atom-node'>
      <item id='70b2a83be71dfca04df91133d953fb22b41b4267'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <source>
            <title>Example Feed</title>
            <link href='http://example.org/'/>
            <link rel='self'
                  type='application/atom+xml'
                  href='http://example.org/atom.xml'/>
            <id>tag:example.org,2003:home</id>
            <updated>2003-12-13T18:30:02Z</updated>
            <author>
              <name>John Doe</name>
            </author>
          </source>
          <title>Atom-Powered Robots Run Amuck</title>
          <summary>Asimov's First Law horribly violated!</summary>
          <link rel='alternate'
                type='text/html'
                href='http://example.org/2003/12/13/atom03'/>
          <id>tag:example.org,2003:entry-32397</id>
          <published>2003-12-13T18:30:02Z</published>
          <updated>2003-12-13T18:30:02Z</updated>
        </entry>
      </item>
     </items>
  </event>
</message>

<message from='pubsub.example.com'
         to='bob@example.com'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='an-atom-node'>
      <item id='70b2a83be71dfca04df91133d953fb22b41b4267'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <source>
            <title>Example Feed</title>
            <link href='http://example.org/'/>
            <link rel='self'
                  type='application/atom+xml'
                  href='http://example.org/atom.xml'/>
            <id>tag:example.org,2003:home</id>
            <updated>2003-12-13T18:30:02Z</updated>
            <author>
              <name>John Doe</name>
            </author>
          </source>
          <title>Atom-Powered Robots Run Amuck</title>
          <summary>Asimov's First Law horribly violated!</summary>
          <link rel='alternate'
                type='text/html'
                href='http://example.org/2003/12/13/atom03'/>
          <id>tag:example.org,2003:entry-32397</id>
          <published>2003-12-13T18:30:02Z</published>
```

```
            <updated>2003-12-13T18:30:02Z</updated>
          </entry>
        </item>
      </items>
    </event>
  </message>
```

2.3. Notification of Entry Modification

TOC

An implementation SHOULD support notifications related to modification of an entry.
When a content author updates an existing dynamic resource, many entities may be interested in learning that the resource has been modified. The process flow is as follows:

- Author updates an existing entry at the HTTP service via the Atom API.

- The HTTP service sends data for the updated Atom entry in an XMPP pubsub "publish" request to a specific node at the XMPP pubsub service, specifying the same Item ID as previously supplied. (Note: If the entry may be copied from one feed to another, e.g., in the generation of "synthetic" feeds, the entry SHOULD contain an atom:source element to ensure consistent metadata.)

- The XMPP pubsub service sends an XMPP message notification to each XMPP entity that is subscribed to the pubsub node.

First the HTTP service sends an XMPP pubsub "publish" request to the XMPP pubsub service (note the modified title and time):

```
<iq type='set'
    from='atompub.example.org'
    to='pubsub.example.com'
    id='publish2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='an-atom-node'>
      <item id='70b2a83be71dfca04df91133d953fb22b41b4267'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <source>
            <title>Example Feed</title>
            <link href='http://example.org/'/>
            <link rel='self'
                  type='application/atom+xml'
                  href='http://example.org/atom.xml'/>
            <id>tag:example.org,2003:home</id>
            <updated>2003-12-13T18:30:02Z</updated>
            <author>
              <name>John Doe</name>
            </author>
          </source>
          <title>Atom-Powered Robots Run Amok</title>
          <summary>Asimov's First Law horribly violated!</summary>
          <link rel='alternate'
                type='text/html'
                href='http://example.org/2003/12/13/atom03'/>
          <id>tag:example.org,2003:entry-32397</id>
          <published>2003-12-13T18:30:02Z</published>
          <updated>2003-12-13T18:31:03Z</updated>
        </entry>
      </item>
    </publish>
  </pubsub>
</iq>
```

Subject to node configuration and/or subscription options, each XMPP subscriber would then receive a pubsub notification, which may or may not contain the Atom payload.

```
<message from='pubsub.example.com'
         to='alice@example.net'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='an-atom-node'>
      <item id='70b2a83be71dfca04df91133d953fb22b41b4267'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <source>
            <title>Example Feed</title>
            <link href='http://example.org/'/>
            <link rel='self'
                  type='application/atom+xml'
                  href='http://example.org/atom.xml'/>
            <id>tag:example.org,2003:home</id>
            <updated>2003-12-13T18:30:02Z</updated>
            <author>
              <name>John Doe</name>
            </author>
          </source>
          <title>Atom-Powered Robots Run Amok</title>
          <summary>Asimov's First Law horribly violated!</summary>
          <link rel='alternate'
                type='text/html'
                href='http://example.org/2003/12/13/atom03'/>
          <id>tag:example.org,2003:entry-32397</id>
          <published>2003-12-13T18:30:02Z</published>
          <updated>2003-12-13T18:31:03Z</updated>
        </entry>
      </item>
     </items>
  </event>
</message>

<message from='pubsub.example.com'
         to='bob@example.com'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='an-atom-node'>
      <item id='70b2a83be71dfca04df91133d953fb22b41b4267'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <source>
            <title>Example Feed</title>
            <link href='http://example.org/'/>
            <link rel='self'
                  type='application/atom+xml'
                  href='http://example.org/atom.xml'/>
            <id>tag:example.org,2003:home</id>
            <updated>2003-12-13T18:30:02Z</updated>
            <author>
              <name>John Doe</name>
            </author>
          </source>
          <title>Atom-Powered Robots Run Amok</title>
          <summary>Asimov's First Law horribly violated!</summary>
          <link rel='alternate'
                type='text/html'
                href='http://example.org/2003/12/13/atom03'/>
          <id>tag:example.org,2003:entry-32397</id>
          <published>2003-12-13T18:30:02Z</published>
```

```
            <updated>2003-12-13T18:31:03Z</updated>
          </entry>
        </item>
      </items>
    </event>
  </message>
```

2.4. Notification of Entry Deletion

TOC

An implementation MAY support notifications related to deletion of an entry.
If a content author deletes an existing dynamic resource, many entities may be interested in learning that the resource is no longer available. The process flow is as follows:

- Author deletes an existing entry at the HTTP service via the Atom API.
- The HTTP service sends an XMPP pubsub "retract" request to a specific node at the XMPP pubsub service, specifying the same Item ID as previously supplied.
- The XMPP pubsub service sends an XMPP message notification to each XMPP entity that is subscribed to the pubsub node.

First the HTTP service sends an XMPP pubsub "retract" request to the XMPP pubsub service:

```
<iq type='set'
    from='atompub.example.org'
    to='pubsub.example.com'
    id='delete1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <retract node='an-atom-node'>
      <item id='70b2a83be71dfca04df91133d953fb22b41b4267'/>
    </retract>
  </pubsub>
</iq>
```

Subject to node configuration and/or subscription options, each XMPP subscriber would then receive a pubsub notification that the item was deleted.

```
<message from='pubsub.example.com'
         to='alice@example.net'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='an-atom-node'>
      <retract id='70b2a83be71dfca04df91133d953fb22b41b4267'/>
     </items>
  </event>
</message>

<message from='pubsub.example.com'
         to='bob@example.com'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='an-atom-node'>
      <retract id='70b2a83be71dfca04df91133d953fb22b41b4267'/>
     </items>
  </event>
</message>
```

3. Implementation Notes

TOC

3.1. Association Between User and Pubsub Node

TOC

As explained in [XMPP-PUBSUB] (Millard, P., Saint-Andre, P., and R. Meijer, “Publish-Subscribe,” March 2008.), a notification MAY include an XMPP SHIM Stanza Headers and Internet Metadata [XMPP-SHIM] (Saint-Andre, P. and J. Hildebrand, “Stanza Headers and Internet Metadata (SHIM),” July 2006.) header named "Reply-To" that specifies the JabberID of the publishing entity. Alternatively, as described in [XMPP-PEP] (Saint-Andre, P. and K. Smith, “Personal Eventing via Pubsub,” September 2007.), the XMPP server of the publishing entity MAY enable the publishing entity to associate a virtual pubsub service with the JabberID of its account, obviating the need for a separate pubsub service.

3.2. Generation of ItemIDs

TOC

The pubsub ItemIDs MUST conform to the rules defined in [XMPP-PUBSUB] (Millard, P., Saint-Andre, P., and R. Meijer, “Publish-Subscribe,” March 2008.). One possible method for generating a unique ItemID is to concatenate the XMPP address of the pubsub service, the pubsub node to which the item is published, and the atom:id of the feed entry, then hash the resulting string using the [SHA1] (Eastlake, D. and P. Jones, “US Secure Hash Algorithm 1 (SHA1),” September 2001.) algorithm.

3.3. Handling of Duplicate Entries

TOC

It is the responsibility of the receiving application to remove or ignore duplicate entries that might be received from multiple feeds.

3.4. Notifications Matching Multiple Subscriptions

TOC

An XMPP entity may subscribe to a publish-subscribe node multiple times (e.g., once for each of several keywords), in which case a single notification may match one or more subscriptions. In order to specify which of one or more subscriptions are matched, the notification message SHOULD specify the subscription IDs using the header syntax defined in [XMPP-SHIM] (Saint-Andre, P. and J. Hildebrand, "Stanza Headers and Internet Metadata (SHIM)," July 2006.) and the "pubsub#subid" header defined in [XMPP-PUBSUB] (Millard, P., Saint-Andre, P., and R. Meijer, "Publish-Subscribe," March 2008.), as shown at the end of the following example.

```
<message from='pubsub.example.com'
         to='alice@example.net'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='an-atom-node'>
      <item id='70b2a83be71dfca04df91133d953fb22b41b4267'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <source>
            <title>Example Feed</title>
            <link href='http://example.org/'/>
            <link rel='self'
                  type='application/atom+xml'
                  href='http://example.org/atom.xml'/>
            <id>tag:example.org,2003:home</id>
            <updated>2003-12-13T18:30:02Z</updated>
            <author>
              <name>John Doe</name>
            </author>
          </source>
          <title>Atom-Powered Robots Run Amok</title>
          <summary>Asimov's First Law horribly violated!</summary>
          <link rel='alternate'
                type='text/html'
                href='http://example.org/2003/12/13/atom03'/>
          <id>tag:example.org,2003:entry-32397</id>
          <published>2003-12-13T18:30:02Z</published>
          <updated>2003-12-13T18:31:03Z</updated>
        </entry>
      </item>
     </items>
  </event>
  <headers xmlns='http://jabber.org/protocol/shim'>
    <header name='pubsub#subid'>123-abc</header>
    <header name='pubsub#subid'>004-yyy</header>
  </headers>
</message>
```

4. Security Considerations

TOC

Detailed security considerations for the relevant protocols profiled in this memo are given in [ATOM-FORMAT] (Nottingham, M. and R. Sayre, "The Atom Syndication Format," December 2005.), [XMPP-CORE] (Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core," October 2004.), and [XMPP-PUBSUB] (Millard, P., Saint-Andre, P., and R. Meijer, "Publish-Subscribe," March 2008.); this memo introduces no new security concerns above and beyond those described in the foregoing specifications.

5. References

TOC

5.1. Normative References

TOC

[ATOM-FORMAT]	Nottingham, M. and R. Sayre, "The Atom Syndication Format," RFC 4287, December 2005 (TXT).
[TERMS]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997.
[XMPP-CORE]	Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core," RFC 3920, October 2004 (TXT).
[XMPP-PUBSUB]	Millard, P., Saint-Andre, P., and R. Meijer, "Publish-Subscribe," XSF XEP 0060, March 2008.

5.2. Informative References

TOC

[ATOM-PROTOCOL]	Gregorio, J. and B. de hOra, "The Atom Publishing Protocol," RFC 5023, October 2007 (TXT).
[HTTP]	Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," RFC 2616, June 1999 (TXT, PS, PDF, HTML, XML).
[SHA1]	Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," RFC 3174, September 2001 (TXT).
[SMTP]	Klensin, J., "Simple Mail Transfer Protocol," RFC 2821, April 2001 (TXT).
[XMPP-PEP]	Saint-Andre, P. and K. Smith, "Personal Eventing via Pubsub," XSF XEP 0163, September 2007.
[XMPP-SHIM]	Saint-Andre, P. and J. Hildebrand, "Stanza Headers and Internet Metadata (SHIM)," XSF XEP 0131, July 2006.

Authors' Addresses

TOC

	Peter Saint-Andre
	XMPP Standards Foundation
Email:	stpeter@jabber.org
URI:	https://stpeter.im/
	Joe Hildebrand
	Jabber, Inc.
Email:	jhildebrand@jabber.com
	Bob Wyman
Email:	bob@wyman.us

Full Copyright Statement

TOC

Intellectual Property