XEP-0470: Pubsub Attachments

Jérôme Poisson
mailto:goffi@goffi.org
xmpp:goffi@jabber.fr
https://www.goffi.org

2022-08-25
Version 0.2.0

Status	Type	Short Name
Experimental	Standards Track	pubsub-attachments

This specification provides a way to attach elements to a pubsub item.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

It is nowadays common to attach informations to messages or other items in various social networks: famous example are ”like” (or ”favourite”) and ”reactions” features.
While there are ways to attach informations to <message/> stanzas with extensions such as Message Fastening (XEP-0422) [1] or Message Reactions (XEP-0444) [2], this is not the case for pubsub items
Some software use comments as a work-around for Microblogging Over XMPP (XEP-0277) [3], by posting a single ”□” character to notify a ”like”. This has the advantage to work out of the box even if no specific implementation is done to manage this, but this has a couple of disadvantages:

- it only works with Microblogging Over XMPP (XEP-0277) [4], it is not possible to like other kind of items;
- it’s polluting comments with an information which should be separated;
- it doesn’t handle uniqueness: a ”like” should be doable only once per entity, but by using comments one can like thousand of times, and it’s the receiving client which must ignore duplicates;
- it doesn’t scale: if thousand of people like a blog post, all comments must be retrieved and counted;
- it’s mixing metadata and content intended for human user;
- this behaviour is found in the wild, but not standardized anywhere;

This XEP proposes an alternative and generic solution, which can work with any kind of pubsub item.

2 Requirements

The design goal of this XEP are:

- work with any kind of pubsub item, not only Microblogging Over XMPP (XEP-0277) [5]
- handle uniqueness of attachment per JID
- have an extensible mechanism for future use

[1] XEP-0422: Message Fastening <https://xmpp.org/extensions/xep-0422.html>.
[2] XEP-0444: Message Reactions <https://xmpp.org/extensions/xep-0444.html>.
[3] XEP-0277: Microblogging over XMPP <https://xmpp.org/extensions/xep-0277.html>.
[4] XEP-0277: Microblogging over XMPP <https://xmpp.org/extensions/xep-0277.html>.
[5] XEP-0277: Microblogging over XMPP <https://xmpp.org/extensions/xep-0277.html>.

- re-use pubsub subscription and access control mechanism
- suitable to implement feature similar to commonly seen ”like/favourite” and ”reactions”.
- optionally have a way to ”group” or ”summarize” informations: get a summary of all attachment without needing to retrieve each of them individually.

To facilitate the bootstrapping of this XEP, it is also designed to work in a basic way with generic pubsub service. However, some implementation work is necessary to offer the full potential of the XEP (and notably to be able to scale).

3 Glossary

- **like/favourite**: a common way to indicate interest in item
- **reactions**: attaching one or more emoji(s) to an item
- **attachment node**: node where attached data of an item are published
- **summary node**: node managed by pubsub service which keep a summary of all attachment of target items
- **target item**: item to which metadata is attached
- **target node**: pubsub node where target items are published
- **attachment item**: item of the attachment node containing attachments by a specific JID
- **summary item**: item of the summary node linked to a target item
- **attachment**: child element of the <attachments> element, describing a metadata attached to a target item

4 Use Cases

4.1 Basic Usage

Romeo wants to indicate to Juliet that he has noticed her post about the balcony restoration. This Microblogging Over XMPP (XEP-0277) [6] item has been published on the PEP service of Juliet at service juliet@capulet.lit on the node ’urn:xmpp:microblog:0’ and the item has the ID ’balcony-restoration-afd1’.
To do so he publishes the following item to the suitable attachment node:

[6] XEP-0277: Microblogging over XMPP <https://xmpp.org/extensions/xep-0277.html>.

Listing 1: Romeo Indicates To Juliet That He Has Noticed Her Publication

```
<iq from='romeo@montague.lit/123'
    id='attachment_1'
    to='juliet@capulet.lit'
    type='set'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:pubsub-attachments:1/xmpp:juliet@capulet.
       lit?;node=urn%3Axmpp%3Amicroblog%3A0;item=balcony-restoration-
       afd1'>
      <item id='romeo@montague.lit'>
        <attachments>
          <noticed timestamp="2022-07-11T12:07:24Z" />
        </attachments>
      </item>
    </publish>
  </pubsub>
</iq>
```

Few seconds later, Romeo reacts with some emojis, it does that with the following item, and his client takes care of keeping the <noticed> element above:

Listing 2: Romeo Add Reactions To Juliet Publication

```
<iq from='romeo@montague.lit/123'
    id='attachment_2'
    to='juliet@capulet.lit'
    type='set'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:pubsub-attachments:1/xmpp:juliet@capulet.
       lit?;node=urn%3Axmpp%3Amicroblog%3A0;item=balcony-restoration-
       afd1'>
      <item id='romeo@montague.lit'>
        <attachments>
          <noticed timestamp="2022-07-11T12:07:24Z" />
          <reactions timestamp="2022-07-11T12:07:48Z">
            <reaction> </reaction>
            <reaction> </reaction>
          </reactions>
        </attachments>
      </item>
    </publish>
  </pubsub>
</iq>
```

4.1.1 Explanations

To attach metadata to a pubsub item, an "attachment node" MAY be created, either by the publisher of the target item, or by the pubsub service if it is fully-compliant with this XEP (see

below). This node name is generated by merging the following strings:

- the namespace '**urn:xmpp:pubsub-attachments:1**'
- a slash "**/**"
- the **XMPP URI** of the target item as explained at XEP-0060 § Pubsub URIs

Thus, in the example above, the node name to use for the item "balcony-restoration-afd1" of the node "urn:xmpp:microblog:0" located at PEP service "juliet@capulet.lit" is: "urn:xmpp:pubsub-attachments:1/xmpp:juliet@capulet.lit?;node=urn%3Axmpp%3Amicroblog%3A0;item=balcony-restoration-afd1"
This node SHOULD have the same access model than the target node.
To publish to this node, an entity MUST use its own bare JID for the ID of the item. It is both to keep the uniqueness of the item per JID and to make the retrieval of attachment for a particular entity easy.
The entity willing to publish attachment tries directly to publish to the above mentioned node. If the node doesn't exist (and is not created on the fly by the pubsub service, see below), the pubsub service SHOULD answer with <item-not-found> error as explained in XEP-0060 §7.1.3.3 Node Does Not Exist. If the node doesn't not exist, that means that it's not possible to attach metadata to the target item, the entity willing to publish the attachment MUST NOT try to create the node itself (that would result in wrong ownership of the node).
An attachment payload is build with a top level <attachments> element which has zero, one or more child elements. This specification defines 2 child elements, <noticed> and <reactions>, but future XEPs may add their own elements qualified by their own namespaces to extend the functionalities. Each child element MAY have an optional 'timestamp' attribute indicating when the element has been attached. The value of this attribute is a DateTime as specified in XMPP Date and Time Profiles (XEP-0082) [7].
Because there is one item per JID; to update, add or remove attachments an entity simply re-publish an item on the same node with its bare JID as ID. It is the responsibility of the publishing entity to republish all previously existing attachments (except those who need to be removed). If an XMPP client doesn't know a specific attachment, it MUST keep it and republish it when updating attachments.
All attachments of a specific JID can be deleted at once by retracting the item as specified at XEP-0060 §7.2 Delete an Item from a Node. A client SHOULD NOT retract an attachment item if there are attachments it doesn't know, instead it SHOULD publish a new attachment item without the attachments which must be removed, and with the unknown attachments left in place.

4.2 Full-Compliance

Previous section describes the basic usage of pubsub attachments, which works with generic pubsub service. However, even if it works out of the box, it relies on the goodwill of entities:

[7] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.

an attacker or simply a bugged implementation could publish an item with wrong ID or somebody else bare JID, an item publisher client could miss the creation of attachment node, or give it bad access model, access model between attachment node and target node can become out of sync, etc.
To avoid these flaws, a pubsub service SHOULD implement the features described in this and following sections. If a pubsub service does so, it is said to be fully-compliant with pubsub attachments, and then and only then it can advertises the feature with Service Discovery (XEP-0030) [8]
To be fully compliant, a PEP or pubsub service MUST implement the following features, which are explained in details below:

- auto-create attachment node, and keep its publish_model and access_model synchronized
- forbid manual creation of attachment or summary node
- check validity of items published to attachment node, and notably the item ID
- create and maintain a summary node
- handle <noticed> and <reactions> attachments

4.3 Automatic Node Creation

When an attachments item is published to a fully-compliant pubsub service, and if the attachment node doesn't exist, the service MUST create automatically the node as explained at XEP-0060 §7.1.4 Automatic Node Creation, except that instead of applying the default configuration, it MUST apply the same access_model and publish_model as for the target node. The service MAY also copy other configuration options if they differ from default, it is up to the implementation to decide which other options are relevant to copy.
If the <iq/> stanza of the publishing client includes publishing options as explained in XEP-0060 § 7.1.5 Publishing Options, they are ignored.
If later the target node configuration is updated and either access_model or pubsub_model are modified, the fully-compliant service MUST also update the attachment node pubsub_model and access_model accordingly.

4.4 Manual Node Creation Rejection

If any user, including owner of target node or publisher of target item, tries to create manually an attachment node or a summary node, a fully-compliant service MUST reject it by returning a <not-allowed/> error.
A client can see if a node creation is necessary by using Service Discovery (XEP-0030) [9]: the

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

presence of 'urn:xmpp:pubsub-attachments:1' feature in *disco#info* means that the service is fully-compliant, and that manual node creation MUST NOT be done.

4.5 Checking Validity of Attachments Items

When an entity publish an items with attachments to an attachment node, a fully-compliant service MUST check that the item is valid by

1. Verifying that the item ID is equal to the bare jid of the item publisher
2. Verifying that the root element of the payload is an <attachments> element qualified by the 'urn:xmpp:pubsub-attachments:1' namespace

If any of these points are not met, the service MUST reject the item by returning a <bad-request/> error.
In addition to those 2 mandatory checks, a pubsub service MAY add implementation specific checks.

4.6 Summary Node

As soon as a first attachment is received, a fully-compliant pubsub service MUST create a "summary node". A summary node is a node maintained by the service which group all attachments of a kind, allowing client to have a good overview of the data without needing to retrieve individually all items of the attachment nodes of all target items.
A summary node has the same access_model as the attachment node, but nobody is allowed to publish directly to it. The summary node is linked to the target node, and its name is made by joining the following element:

1. the '**urn:xmpp:pubsub-attachments:summary:1**' prefix
2. a slash "**/**"
3. the **name of the target node**

Thus in the initial example, for the blog of Juliet, the summary node name would be 'urn:xmpp:pubsub-attachments:summary:1/urn:xmpp:microblog:0' and it would be located at the PEP service juliet@capulet.lit.
For each item of the target node which has attachments, the summary node MUST contain an item which MUST have the same ID. This item contain a <summary> element qualified with the namespace 'urn:xmpp:pubsub-attachments:summary:1'. This item has elements with names matching attachments elements names, and a summary data which depend of the attachment. This specifications explain below how to summarize <noticed> and <reactions> attachments, it is the up to other XEPs specifying other features to explain how to summarize

their own attachments. If a service doesn't know how to summarize an attachment, it SHOULD ignore it.
If a target item has no attachment at all, or if all attachments have been removed, the node MAY either return an <item-not-found> error, or an empty <summary> element, whatever is simpler for the service implementation.
Summary node subscriptions are working as for normal pubsub nodes: when a new attachment is published, resulting in the corresponding summary item updated, an event is sent with the new item to every subscribers.

Listing 3: Romeo Check Summary of Attachments of Juliet Blog

```
<iq from='romeo@montague.lit/123'
    id='attachment_3'
    to='juliet@capulet.lit'
    type='get'>
    <pubsub xmlns='http://jabber.org/protocol/pubsub'>
      <items node='urn:xmpp:pubsub-attachments:summary:1/
          urn:xmpp:microblog:0' />
    </pubsub>
  </iq>
```

Listing 4: Fully-Compliant Pubsub Service Returns Summary Items

```
<iq from='juliet@capulet.lit'
    id='attachment_3'
    to='romeo@montague.lit/123'
    type='result'>
    <pubsub xmlns='http://jabber.org/protocol/pubsub'>
      <items node='urn:xmpp:pubsub-attachments:summary:1/
          urn:xmpp:microblog:0'>
        <item id='balcony-restoration-afd1'>
          <summary xmlns='urn:xmpp:pubsub-attachments:summary:1'>
            <noticed count="5" />
            <reactions>
              <reaction count="2"> </reaction>
              <reaction> </reaction>
              <reaction> </reaction>
              <reaction> </reaction>
            </reactions>
          </summary>
        </item>
        <!-{}- ... -{}->
      </items>
    </pubsub>
  </iq>
```

4.7 Noticed Attachment

4.7.1 Foreword: ”noticed” instead of ”like” or ”favourite”

The <noticed> feature described here is similar to what is most often known as ”like”, and sometime ”favourite”. It has been decided to use ”noticed” word to highlight a different spirit from its ancestors.
The ”like” feature has been invented in mid 2000s on commercial social network. Over the years, this functionality has proven to be a borderline toxic problem. Among known issues, we can mention:

- It may cause addictive behaviour, people feeling need to get more ”likes”.
- In contrast, the lack of like on a publication may lead to feelings of depression.
- It is used as a marketing tool, to spy user tastes and interests. It can even be used to discover political orientation, sexual preferences or religious beliefs, which can be dangerous in some countries/locations.
- It tend to diminish the quality of contents, by favoring metrics over contents themselves.
- In some social networks, more likes means more visibility and having a better image, resulting in some people/organizations/companies buying fake likes.
- The word ”like” is ill-suited to bad news or dramatics events, when someone simply wants to show their support or empathy.

For all these reasons, it has been decided to use the word ”noticed” which reflect better the way it is used by some people (notably observed on some social network built on top of the ActivityPub protocol): it is then used as way to say ”I have seen” or ”I’ve taken that into account”.
However, and for compatibly reason with other protocols (especially to have the tools to make gateways), the summary feature of <noticed> attachment does count the number of elements. After reading this note, it is up to the various implementations to decide whether to show this number prominently, inconspicuously, or not at all.

4.7.2 Attachment Overview

<noticed> element is attached by an entity to say that they have seen or taken into account something. On the client UI side, it is often published when user push a simple button or icon, and the attachment is often visible with the same icon displayed on the noticed item. If an icon is used, it is recommended to use something as neutral as possible, thus a heart icon SHOULD NOT be used to avoid misunderstanding between various implementations (also see foreword above). As for any attachment, an optional ”timestamp” attribute MAY be set with a

value of latest publication DateTime as specified in XMPP Date and Time Profiles (XEP-0082) [10].

4.7.3 Summarizing

To summarize <noticed> attachments, a fully-compliant pubsub service just sum-up the total number of <noticed> elements found for the item, and put this number in "count" attribute of the summary <noticed> element. In the example below, an item has been noticed 25 times.

Listing 5: Example of Noticed Attachment Summary

```
<iq from='pubsub.example.net'
    id='attachment_4'
    to='juliet@capulet.lit/123'
    type='result'>
    <pubsub xmlns='http://jabber.org/protocol/pubsub'>
      <items node='urn:xmpp:pubsub-attachments:summary:1/
          urn:xmpp:example:0'>
        <item id='ball-event-ab1e'>
          <summary xmlns='urn:xmpp:pubsub-attachments:summary:1'>
            <noticed count="25" />
          </summary>
        </item>
      </items>
    </pubsub>
  </iq>
```

4.8 Reactions Attachment

4.8.1 Attachment Overview

<reactions> element lets an entity attach various emojis to an item. Each emoji is put as the content of a single <reaction> element, and a client SHOULD ensure that any <reaction> element only appears once at most. As for any attachment, a "timestamp" attribute may be set with the DateTime of latest publication to the root <reactions> element. The protocol is similar to Message Reactions (XEP-0444) [11] which is used for <message/> stanza.

4.8.2 Summarizing

To summarize <reactions> attachments, a fully-compliant pubsub service counts how many times each emoji is attached, ignoring duplicate from the same JID if any. If an emoji appears multiple times (from distinct bare JIDs), a 'count' attribute MUST be added to the <reaction> element with the number of time this reaction appear in all reactions as a value (if the same

[10] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.
[11] XEP-0444: Message Reactions <https://xmpp.org/extensions/xep-0444.html>.

reaction appears several times for a single bare JID, it MUST be counted only once).
In following example, all emojis are attached only once to the item, except the woman dancing one which appears 22 times and the ballet shoes one which appears twice.

Listing 6: Example of reactions Attachment Summary

```
<iq from='pubsub.example.net'
    id='attachment_5'
    to='juliet@capulet.lit/123'
    type='result'>
    <pubsub xmlns='http://jabber.org/protocol/pubsub'>
      <items node='urn:xmpp:pubsub-attachments:summary:1/
          urn:xmpp:example:0'>
        <item id='ball-event-ab1e'>
          <summary xmlns='urn:xmpp:pubsub-attachments:summary:1'>
            <reactions>
              <reaction count="22"> </reaction>
              <reaction count="2"> </reaction>
              <reaction> </reaction>
              <reaction> </reaction>
              <reaction> </reaction>
            </reactions>
          </summary>
        </item>
      </items>
    </pubsub>
  </iq>
```

5 Business Rules

- Similarly to "like" in commercial software, the "noticed" attachment can be used to analyse user's tastes, political view, religious beliefs, sexual orientation, etc. It is recommended that implementers post a prominent notice warning users of potential abuses.
- Emoji pictures may differ widely on various platforms where they are displayed. This has already led to misunderstanding of reactions, as a slightly different picture can be interpreted in a completely different way from what the reactions author meant. Here again, a prominent notice in implementations warning user is recommended.
- As "reactions" attachment is similar to Message Reactions (XEP-0444) [12] which is used for <message/> stanza, non <message/> related business rules from there apply for this attachment too. Notably: *A <reaction> element SHOULD only contain Unicode codepoints that can be displayed as a single emoji, as specified in the latest revision of the Unicode Technical Standard #51 [13]. Receiving entities MAY ignore <reaction> elements that do not comply with this*

[12] XEP-0444: Message Reactions <https://xmpp.org/extensions/xep-0444.html>.
[13] Unicode Technical Standard #51 <http://www.unicode.org/reports/tr51/>.

specification.

6 discovering support

If and only if a PEP or pubsub service is fully-compliant with the "Pubsub Attachments" protocol (as explained in Full-Compliance section), it MUST advertise that fact by including the "urn:xmpp:pubsub-attachments:1" discovery feature in response to a Service Discovery (XEP-0030) [14] 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:pubsub-attachments:1'/>…

  </query>
</iq>
```

7 Security Considerations

TODO

8 IANA Considerations

TODO

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

9 XMPP Registrar Considerations

TODO

10 XML Schema

TODO