XEP-0460: Pubsub Caching Hints

Jérôme Poisson
mailto:goffi@goffi.org
xmpp:goffi@jabber.fr

2021-08-10
Version 0.1.0

Status	Type	Short Name
Experimental	Standards Track	pubsub-caching

This specification provides a way to get caching information from a Pubsub node

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction 1

2 Requirements 1

3 Glossary 1

4 Caching Hints Discovery 2

5 Node Persistence 2

6 Max Items and Item Expire 2

7 Consistent Items: items are always identical for all users 3

8 Consistent Set: All Users Have the Same Series of Items 3

9 Stable Items: Items Won't Appear Out of Order 3

10 Always Notify 4

11 Public Node 4

12 The Node can be used for Suggestions 5

13 Purging a Node 5

14 Summary 5

15 Example 7

16 discovering support 8

17 Implementation Notes 9

18 Security Considerations 9

19 IANA Considerations 9

20 XMPP Registrar Considerations 9

21 XML Schema 9

1 Introduction

XMPP Pubsub as described in Publish-Subscribe (XEP-0060) [1] is a very powerful and versatile tool, which is used for numerous XMPP features. For many reasons, notably speed improvement and resources optimisation, XMPP clients may want to cache internally Pubsub nodes and keep cache synchronised with cached pubsub service. Unfortunately the flexibility of Pubsub makes the choice of a good caching strategy complicated and non optimal. This XEP standardize a way for the Pubsub service to give extra information to fix this situation.

2 Requirements

Caching information must be using base XEP-0060 Pubsub features and be easy to obtain for the client, and easy to add for the pubsub service. The desired goals are:

- use existing XEP-0060 features to get the data
- avoid duplication of data in cache
- know if cache can be shared between users
- know if a data can be re-used in a ”discovery” feature
- know if items are silently removed or modified
- know if data synchronisation notifications (new items, deletion) are always sent

3 Glossary

- **Dynamic Items** — items which may vary according to parameters like requesting user or time of the day
- **Static Items** — items which don’t change dynamically. This is the most common case.
- **Consistent Set** — set of items inside a node is a same whatever allowed user is requesting it.
- **Stable Items** — items are added in order (to the end of the queue).
- **Silently** — in this context, silently means that a modification happens to an item without sending notification to subscribers.

[1] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

4 Caching Hints Discovery

Pubsub Caching Hints are using the Pubsub node metadata as described in Discover Node Metadata section of Publish-Subscribe (XEP-0060) [2]
Hints are advertised using well known fields in Pubsub metadata disco extension. If a Pubsub service implements this XEP, and if it also manages a Personal Eventing Protocol (XEP-0163) [3] service, the fields described here MUST be present for both Pubsub and PEP nodes.

5 Node Persistence

To properly cache a node, a client must known if they are kept in storage or not. To advertise that fact, a Pubsub node compliant with this standard MUST use a field named {urn:xmpp:pubsub-caching:0}persistence of type list-single and whose value can be one of:

- **persistent** — items are kept in persistent storage
- **semi-persistent** — items are kept in temporary storage (e.g. memory storage), and may disappear without notification (e.g. the server is restarted)
- **transient** — items are not stored, and are only sent to subscribers once published (but they can't be retrieve with a Pubsub Get)
- **transient-with-last-item** — only last item is kept in cache and can be retrieved by a Pubsub Get

6 Max Items and Item Expire

It may be necessary to cache a node, to know how many items the Pubsub Service is keeping before silently deleting them, or when they do expire. This is done by advertising it using the pubsub#max_items and pubsub#item_expire fields of type integer-or-max (this type is defined in XEP-0060).
Both fields are mentioned in XEP-0060, but as a reminder:

- **pubsub#max_items** — indicates the maximum number of items that are kept in a node. Above this number items are silently removed. max indicate that server limit is used (if this limit is known, a Pubsub implementation SHOULD indicated it explicitly).
- **pubsub#item_expire** — indicates the number of seconds before an item is silently removed. max is used if there is not limit

[2] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[3] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.

7 Consistent Items: items are always identical for all users

Pubsub is most often used to let allowed users store and retrieve unmodified items. However, nothing in Publish-Subscribe (XEP-0060) [4] prevents a Pubsub service to return dynamic items with identical IDs depending on factors like the JID of the requestor, time of the day, or something else. For instance, a weather service could return local prevision for following day using item identifier like "tomorrow_forecast", or a machine learning algorithm could return favorite Shakespeare books to a user using "favorite" item id, in which case romeo@montaigu.lit would probably have different results than juliet@capulet.lit
For obvious reason those items are hard to impossible to cache. Pubsub services SHOULD avoid using dynamic items, unless there is a really good reason for it.
If a Pubsub service implements Caching Hints and if items are static (i.e. items with the same ID are identical whatever user is requesting them, time of day it is, or any other variable parameter), then it MUST advertise this fact by using the field named {urn:xmpp:pubsub-caching:0}consistent-items of type boolean with the value of true.
Note that overwriting an items as specified in XEP-0060 Publish an Item to a Node is a normal Pubsub use case which SHOULD result in proper notifications being sent to subscribers, if the item is not otherwise different, this is considered as static item and MUST result in a value of true for the {urn:xmpp:pubsub-caching:0}consistent-items field.
Respectively, if a Pubsub node delivers one or more dynamic items, it MUST avertise the fact by using the value of false for the same field.

8 Consistent Set: All Users Have the Same Series of Items

A Pubsub service may have a feature to restrict individual items from a node to some entities (e.g. to have some items only visible to family, friends or coworkers). In this case, we say that the node has inconsistent items set, and this implies that cache must not be shared between users (as some users may have access to some items that other don't).
If a Pubsub node is always returning the same items ids to all allowed users, it MUST advertise this fact by using the value true for the boolean field {urn:xmpp:pubsub-caching:0}consistent-set.
On the other hand, if a Pubsub node may return different items according to the requesting entities (assuming that entities are allowed at the node level), it MUST advertise this fact by using the value false for the same field.

9 Stable Items: Items Won't Appear Out of Order

Normally, items are managed like a queue in a node, i.e. new items are appended to one ends, and existing items can only be deleted (or overwritten, in which case an item with the same ID

[4] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

is appended to the end). However, it may be necessary for a Pubsub service to include items out of order (i.e. not appending it at the end), for instance when a Pubsub service is a bridge to a third party protocol which receives items out of order.
Unstable items doesn't change the fact that node can be shared between users or node, but it may have an impact on client implementation, as the caching implementation may have items in a different order, and items may be missed, thus this fact is valuable to know for a client willing to cache the node. Note that unstable items SHOULD be avoided by a Pubsub service whenever it's possible.
If items are always appended to the end of the queue, the Pubsub node MUST advertise this fact using the {urn:xmpp:pubsub-caching:0}stable-items field of type boolean with a value of true.
On the other hand, if items order can't be guaranteed, the Pubsub node MUST advertise this fact by using the value of false for the same field

10 Always Notify

To synchronize correctly a Pubsub node, an XMPP client must be aware of any modification that happen to its items or to the node itself. This is possible thanks to the subscription mechanism of Publish-Subscribe (XEP-0060) [5]. However, notification can be skipped, notably item retraction notification must be explicitly requested by the client, and thus may be missing, resulting in cache becoming out of sync with the Pubsub service
To avoid that, the Pubsub service may enforce notifications for all modifying events to a node or its items, even if they are not explicitly requested by the user doing the modification.
If a node always sends notification, including <retact> notifications even if *notify* attribute is not set, then it must advertise this fact using the field named {urn:xmpp:pubsub-caching:0}alway-notify of type boolean with the value of true. If notifications may be omitted, then the same field must be used with the value of false.
A Pubsub service allowing a node to have notifications always sent SHOULD allow the node owners to activate or deactivate this feature through node configuration, using the well-known field with the same name of {urn:xmpp:pubsub-caching:0}always-notify and the same type of boolean. It's up to the implementation to determine if the default value should be true or false.

11 Public Node

If the node is public, i.e. if it as an open access model that means that a client can safely share the cache between users (providing that the consistent items and consistent set fields are also both true).
If a Pubsub node is public, it MUST advertise this fact by exposing it in its metadata using the

[5] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

field named pubsub#access_model of type list-single with a value of open
If the node has an other access model, it is up to the Pubsub implementation to advertise publicly or not this data in the node metadata. It may be a privacy concern to expose any other access model than open

12 The Node can be used for Suggestions

An XMPP client may have a feature to suggest new Pubsub nodes to users (for network exploration, and let users find rapidly interesting content). If the discovery feature is not restricted to some users somehow, this SHOULD be done using only public nodes. But even for a public node, the node owner may no be willing to have they node suggested to random users.
To avoid using inappropriately a public node for suggestion, a Pubsub node MUST announce if the node is usable for suggestion or node by using the field named {urn:xmpp:pubsub-caching:0}allowed-for-suggestions of type boolean. It is up to the Pubsub implementation to decide how this field is set, but it SHOULD have a default value of false and it should be modifiable by node owner through node configuration, using the well-known field with the same name of {urn:xmpp:pubsub-caching:0}allowed-for-suggestions and the same type of boolean.
A client MUST NOT use a node with {urn:xmpp:pubsub-caching:0}allowed-for-suggestions set to false for suggestions

13 Purging a Node

Publish-Subscribe (XEP-0060) [6] wording about purging all node items is not clear about the last item, and it may or may not be kept.
To make it explicit, a client implementing this specification MUST use the field named {urn:xmpp:pubsub-caching:0}purge-keep-last-item of type boolean with the value of true if the last item is NOT retracted when a node purge is performed. On the opposite, the value of true MUST be used if ALL items are retracted when a node purge is performed, actually leaving the node empty, with no item at all.

14 Summary

Here a is a table summarizing all fields to announce when implementing this XEP. All fields but pubsub#access_model are mandatory if a Pubsub service advertise support for this XEP.

[6] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

name	field	type	meaning	comment
Max Items	pubsub#max_-items	integer-or-max	How many items are kept in storage	
Item Expire	pubsub#item_-expire	integer-or-max	How many seconds items are kept	
Node Persistence	{urn:xmpp:pubsub-caching:0}persistence	list-single	Items are kept in persistent storage	
Consistent Items	{urn:xmpp:pubsub-caching:0}consistent-items	boolean	Items are static	
Consistent Set	{urn:xmpp:pubsub-caching:0}consistent-set	boolean	All users have the same items	
Stable Items	{urn:xmpp:pubsub-caching:0}stable-items	boolean	Items are not added out of order	
Always Notify	{urn:xmpp:pubsub-caching:0}always-notify	boolean	Modifying Notifications are Always sent to subscribers, even if not explicitly requested by publisher.	
Public Node	pubsub#access_-model	list-single	Items can be retrieved by anybody	if value is not open, it may be omitted
Allowed for Suggestions	{urn:xmpp:pubsub-caching:0}allowed-for-suggestions	boolean	Node and its items can be suggested to random users	SHOULD be settable by node owner, and MUST default to false
Purge Keep Last Item	{urn:xmpp:pubsub-caching:0}purge-keep-last-item	boolean	Last item is always kept when a node purge is performed	

15 Example

Here is an example of Pubsub metadata advertised by a node on a service implementing this XEP. This example is a ”happy path”, i.e. features announced here are cache friendly.

Listing 1: Entity queries a node for information

```
<iq type='get'
    from='francisco@denmark.lit/barracks'
    to='pubsub.shakespeare.lit'
    id='meta1'>
  <query xmlns='http://jabber.org/protocol/disco#info'
         node='princely_musings'/>
</iq>
```

Listing 2: Entities Receive Pubsub Node Metadata with Caching Hints

```
<iq type='result'
    from='pubsub.shakespeare.lit'
    to='francisco@denmark.lit/barracks'
    id='caching_hints'>
  <query xmlns='http://jabber.org/protocol/disco#info'
         node='princely_musings'>
    <identity category='pubsub' type='leaf'/>
    <feature var='http://jabber.org/protocol/pubsub'/>
    <x xmlns='jabber:x:data' type='result'>
      <field var='pubsub#title' label='A␣short␣name␣for␣the␣node' type
          ='text-single'>
        <value>Princely Musings (Atom)</value>
      </field>
      <field var='pubsub#max_items' label='How␣many␣items␣are␣kept␣in␣
          storage' type='text-single'>
        <value>max</value>
      </field>
      <field var='pubsub#item_expire' label='How␣many␣seconds␣items␣
          are␣kept' type='text-single'>
        <value>max</value>
      </field>
      <field var='{urn:xmpp:pubsub-caching:0}persistence' label='How␣
          items␣are␣stored' type='text-single'>
        <value>persistent</value>
      </field>
      <field var='{urn:xmpp:pubsub-caching:0}consistent-items' label='
          Are␣items␣static' type='boolean'>
        <value>true</value>
      </field>
      <field var='{urn:xmpp:pubsub-caching:0}consistent-set' label='
          Are␣items␣set␣consistent' type='boolean'>
        <value>true</value>
```

```
      </field>
      <field var='{urn:xmpp:pubsub-caching:0}stable-items' label='Are␣
          items␣stable' type='boolean'>
        <value>true</value>
      </field>
      <field var='{urn:xmpp:pubsub-caching:0}always-notify' label='Are
          ␣notifications␣always␣sent' type='boolean'>
        <value>true</value>
      </field>
      <field var='pubsub#access_model' label='Access␣model' type='list
          -single'>
        <value>open</value>
      </field>
      <field var='{urn:xmpp:pubsub-caching:0}allowed-for-suggestions'
          label='Can␣node␣be␣used␣for␣suggestions' type='boolean'>
        <value>true</value>
      </field>
      <field var='{urn:xmpp:pubsub-caching:0}purge-keep-last-item'
          label='Is␣last␣item␣kept␣when␣a␣node␣purge␣is␣performed'
          type='boolean'>
        <value>false</value>
      </field>
    </x>
  </query>
</iq>
```

16 discovering support

If a server supports the ”Pubsub Caching Hints” protocol, it must advertize it by including the ”urn:xmpp:pubsub-caching:0” discovery feature (see Protocol Namespaces regarding issuance of one or more permanent namespaces) in response to a Service Discovery (XEP-0030) [7] information request:

Listing 3: 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 4: service discovery information response

```
<iq from='example.com'
```

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

```
    id='disco1'
    to='example.org'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>…

    <feature var='urn:xmpp:pubsub-caching:0'/>…

  </query>
</iq>
```

17 Implementation Notes

As order of insertion and overwriting of items may be relevant to the client, it is recommended for caching-friendly Pubsub service to implement Order-By (XEP-0413) [8], thus client can cache items using an order by date of creation.

18 Security Considerations

TODO

19 IANA Considerations

TODO

20 XMPP Registrar Considerations

TODO

21 XML Schema

TODO

[8] XEP-0413: Order-By <https://xmpp.org/extensions/xep-0413.html>.