XEP-0413: Order-By

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

2021-07-21
Version 0.2

Status	Type	Short Name
Experimental	Standards Track	NOT_YET_ASSIGNED

This specification allows to change order of items retrieval in a Pubsub or MAM query

Legal

Copyright

Permissions

Warranty

Liability

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction

Publish-Subscribe (XEP-0060) [1] §6.5.7 allows to retrieve the "most recent items" and Message Archive Management (XEP-0313) [2] state in §3.1 that archives are ordered in "chronological order". While this order is straightforward in general use cases, it is sometimes desirable to use a different order, for instance while using Microblogging Over XMPP (XEP-0277) [3]: a spelling mistake correction should not bring an old blog post to the top of retrieved items. This specification allows to explicitly change business logic to retrieve the items in a different order.

2 Requirements

- an entity should be able to retrieve items by date of creation or by date of last modification (see below for definitions)
- the specification should be extensible to allow new ordering
- in case of conflicts, a 2nd, 3rd, etc. level of ordering should be possible

3 Glossary

In XEP-0060, there is no such thing as "updated item". This XEP changes the business logic as follow:

- **Date of creation** — date when the item has been published **ONLY if the item has a new id** (i.e. an id which was not already present in the node at the time of publication). If an item reuses an existing id, it overwrites the original item **and the date of creation stays the date of creation of the original item**.
- **Date of modification** — date when the item has been overwritten by a new item of the same id. If the item has never been overwritten, it is equal to the date of creation defined above.
- **Order Field** — data used in the by attribute (e.g. creation or modification)

4 Use Cases

4.1 Retrieve Items By Date of Creation

Juliet wants to retrieve plays of her favorite writer, William Shakespeare. She wants to retrieve the 3 most recent ones by date of creation.

[1] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[2] XEP-0313: Message Archive Management <https://xmpp.org/extensions/xep-0313.html>.
[3] XEP-0277: Microblogging over XMPP <https://xmpp.org/extensions/xep-0277.html>.

To do so, her client do a regular Pubsub request, but adds the <order> element as a children of the <pubsub> element with the ”urn:xmpp:order-by:1” namespace, a by attribute equal to creation and a desc attribute equal to true.

Listing 1: Retrieving items ordered by date of creation

```
<iq type='get'
    from='juliet@capulet.lit/balcony'
    to='pubsub.shakespeare.lit'
    id='{}'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='plays' max_items='3'/>
    <order xmlns='urn:xmpp:order-by:1' by='creation' desc='true'/>
  </pubsub>
</iq>
```

The Pubsub service then returns the 3 most recently created plays, first one being the most recent.

Listing 2: Service returns all items

```
<iq type='result'
    from='pubsub.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    id='{}'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='plays'>
      <item id='153214214'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <title>Henry VIII</title>
        </entry>
      </item>
      <item id='623423544'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <title>Tempest</title>
        </entry>
      </item>
      <item id='452432423'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <title>Wintter's Tale</title>
        ␣␣␣␣␣␣␣␣</entry>
      ␣␣␣␣␣␣</item>
    ␣␣␣␣</items>
  ␣␣</pubsub>
</iq>
```

4.2 Retrieve Items By Date of Modification

Juliet realizes that there is a spelling mistake, it's "Winter's Tale" and not "Wintter's Tale". She fixes it by overwritting the item:

Listing 3: Juliet Overwritte the Item to Fix It

```
<iq type='set'
    from='juliet@capulet.lit/balcony'
    to='pubsub.shakespeare.lit'
    id='orderby2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='plays'>
      <item id='452432423'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <title>Winter's Tale</title>
        </entry>
      </item>
    </publish>
  </pubsub>
</iq>
```

To check that everything is alright, she requests again the last 3 items, but this time by date of modification. To do so, the client proceeds the same way as for date of creation, except that it uses the value modification for the by attribute.

Listing 4: Retrieving items ordered by date of modification

```
<iq type='get'
    from='juliet@capulet.lit/balcony'
    to='pubsub.shakespeare.lit'
    id='orderby3'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='plays' max_items='3'/>
    <order xmlns='urn:xmpp:order-by:1' by='modification' desc='true'/>
  </pubsub>
</iq>
```

The Pubsub service returns again the 3 plays but the "Winter Tales" item has been overwritten recently, while the 2 others have never been overwritten, so it returns the items in the following order, with the most recently modified item on top:

Listing 5: Service returns all items

```
<iq type='result'
    from='pubsub.shakespeare.lit'
    to='juliet@capulet.lit/balcony'
    id='orderby3'>
```

```
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='plays'>
      <item id='452432423'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <title>Winter's Tale</title>
        </entry>
      </item>
      <item id='153214214'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <title>Henry VIII</title>
        </entry>
      </item>
      <item id='623423544'>
        <entry xmlns='http://www.w3.org/2005/Atom'>
          <title>Tempest</title>
        </entry>
      </item>
    </items>
  </pubsub>
</iq>
```

4.3 Use with MAM

With Message Archive Management (XEP-0313) [4] the logic is the same, but the <order> element is added as a child of the <query> element:

Listing 6: MAM Pubsub Query with Ordering

```
<iq to='pubsub.shakespeare.lit' type='set' id='orderby4'>
  <query xmlns='urn:xmpp:mam:2' queryid='123' node='plays'>
    <order xmlns='urn:xmpp:order-by:1' by='creation'/>
  </query>
</iq>
```

This way, filters can be used with a specific ordering.

4.4 Reversing the Order

By default, ordering MUST be done in ascending order. This can be reversed by using the desc boolean attribute, which MAY have a value of either true or 1.

[4] XEP-0313: Message Archive Management <https://xmpp.org/extensions/xep-0313.html>.

4.5 Using Order-By with RSM

This section provides a full example of using Order-By with Pubsub and RSM. For readability, we'll use a node with 4 items that will have following IDs (in order of their creation) A, B, C and D. Items C has been overwritten after D creation, and item A has been overwritten even later. Thus, when *ascending* creation order is requested, items are in order A, B, C, D. When *ascending* modification order is requested, items are in order B, D, C, A. Let's see how this work when Juliet wants to retrieve all items in ascending modification order with RSM using a page size of 2 items:

Listing 7: Juliet Retrieves First Page of Items with RSM

```
<iq id="rsm_1" type="get" from="juliet@capulet.lit/123">
    <pubsub xmlns="http://jabber.org/protocol/pubsub">
        <items node="balcony"/>
        <order xmlns="urn:xmpp:order-by:0" by="modification"/>
        <set xmlns="http://jabber.org/protocol/rsm">
            <max>2</max>
        </set>
    </pubsub>
</iq>
```

Listing 8: Pubsub Returns First Page

```
<iq from="ordered_pubsub@capulet.lit" id="rsm_1" to="juliet@capulet.
   lit/123" type="result">
    <pubsub xmlns="http://jabber.org/protocol/pubsub">
        <items node="balcony">
            <item id="B" publisher="romeo@montaigu.lit/456">
                <payload xmlns="http://somenamespace.example.com">
                        item B
                </payload>
            </item>
            <item id="D" publisher="romeo@montaigu.lit/456">
                <payload xmlns="http://somenamespace.example.com">
                        item D
                </payload>
            </item>
        </items>
        <set xmlns="http://jabber.org/protocol/rsm">
            <first index="0">B</first>
            <last>D</last>
            <count>4</count>
        </set>
    </pubsub>
</iq>
```

Now Juliet wants to get the second and last page to complete her collection. She does this as usual with RSM, by using the value advertised in *<last>* element in a *<after>* element.

NOTE: in this example the value used in *<last>* element is the item ID, but as specified in Result Set Management (XEP-0059) [5], an implementation MAY use whatever makes sense to it, the requesting client MUST treat this as an opaque value.

Listing 9: Juliet Retrieves the Second (and Last) Page of Items

```
<iq id="rsm_2" type="get" from="juliet@capulet.lit/123">
    <pubsub xmlns="http://jabber.org/protocol/pubsub">
        <items node="balcony"/>
        <order xmlns="urn:xmpp:order-by:0" by="modification"/>
        <set xmlns="http://jabber.org/protocol/rsm">
            <max>2</max>
            <after>D</after>
        </set>
    </pubsub>
</iq>
```

Listing 10: Pubsub Service Returns Second Page

```
<iq from="ordered_pubsub@capulet.lit" id="rsm_2" to="juliet@capulet.
   lit/123" type="result">
    <pubsub xmlns="http://jabber.org/protocol/pubsub">
        <items node="balcony">
            <item id="C" publisher="romeo@montaigu.lit/456">
                <payload xmlns="http://somenamespace.example.com">
                        item C
                </payload>
            </item>
            <item id="A" publisher="romeo@montaigu.lit/456">
                <payload xmlns="http://somenamespace.example.com">
                        item A
                </payload>
            </item>
        </items>
        <set xmlns="http://jabber.org/protocol/rsm">
            <first index="2">C</first>
            <last>A</last>
            <count>4</count>
        </set>
    </pubsub>
</iq>
```

Juliets wonders which are the 2 last items created. To discover this, she request again the node, but this time with a creation order field, and in descending order:

Listing 11: Juliet Retrieves Last Created Items

[5] XEP-0059: Result Set Management <https://xmpp.org/extensions/xep-0059.html>.

```
<iq id="rsm_3" type="get" from="juliet@capulet.lit/123">
    <pubsub xmlns="http://jabber.org/protocol/pubsub">
        <items node="balcony"/>
        <order xmlns="urn:xmpp:order-by:0" by="creation" desc='true'/>
        <set xmlns="http://jabber.org/protocol/rsm">
            <max>2</max>
        </set>
    </pubsub>
</iq>
```

Listing 12: Pubsub Service Returns Last Created Items

```
<iq from="ordered_pubsub@capulet.lit" id="rsm_3" to="juliet@capulet.
   lit/123" type="result">
    <pubsub xmlns="http://jabber.org/protocol/pubsub">
        <items node="balcony">
            <item id="D" publisher="romeo@montaigu.lit/456">
                <payload xmlns="http://somenamespace.example.com">
                        item D
                </payload>
            </item>
            <item id="C" publisher="romeo@montaigu.lit/456">
                <payload xmlns="http://somenamespace.example.com">
                        item C
                </payload>
            </item>
        </items>
        <set xmlns="http://jabber.org/protocol/rsm">
            <first index="0">D</first>
            <last>C</last>
            <count>4</count>
        </set>
    </pubsub>
</iq>
```

Now she knows that last created item is D, and the one created before is C.
Please note that items are in descending order in the whole result set but also inside the RSM page (thus the first item here is D), and that in this order, this request returns the first page, so index is 0 here.
If Juliet wanted to retrieve the second page of items by descending order of creation, she would do like this:

Listing 13: Juliet Retrieves Second Page of Last Created Items

```
<iq id="rsm_4" type="get" from="juliet@capulet.lit/123">
    <pubsub xmlns="http://jabber.org/protocol/pubsub">
        <items node="balcony"/>
        <order xmlns="urn:xmpp:order-by:0" by="creation" desc="true"/>
        <set xmlns="http://jabber.org/protocol/rsm">
```

```
            <max>2</max>
            <after>C</after>
        </set>
    </pubsub>
</iq>
```

Listing 14: Pubsub Service Returns Second Page of Items Orderded by Descending Creation Date

```
<iq from="ordered_pubsub@capulet.lit" id="rsm_4" to="juliet@capulet.
    lit/123" type="result">
    <pubsub xmlns="http://jabber.org/protocol/pubsub">
        <items node="balcony">
            <item id="B" publisher="romeo@montaigu.lit/456">
                <payload xmlns="http://somenamespace.example.com">
                        item B
                </payload>
            </item>
            <item id="A" publisher="romeo@montaigu.lit/456">
                <payload xmlns="http://somenamespace.example.com">
                        item A
                </payload>
            </item>
        </items>
        <set xmlns="http://jabber.org/protocol/rsm">
            <first index="2">B</first>
            <last>A</last>
            <count>4</count>
        </set>
    </pubsub>
</iq>
```

4.6 Extending this Specification

This specification can be extended by further XEPs, proposing other kind of ordering in the 'by' attribute (e.g. ordering by filename for a file sharing service). But this is beyond the scope of this XEP, and a client should not assume that other ordering than "creation" and "modification" are available without further negotiation. Any new ordering specified in a other XEP SHOULD use the Clark notation to avoid any collision (i.e.: {some_namespace}some_ordering).

5 General Considerations

It is important to note the following points:

- Order-By affect the order of the whole archive, **AND** the order of the items inside a RSM result set (i.e. inside a page).

- The order of creation or modification is the one set by the Pubsub service itself. Some Pubsub based features like Microblogging Over XMPP (XEP-0277) [6] let users specify a creation and modification date; using them would need item parsing and is NOT what creation and modification is referring to here. A future XEP extending this one could allow to order by user-specified creation or modification date, but this is beyond the scope of this XEP.
- The semantic described here can be reused in other use cases as for Pubsub or MAM. If it is the case, the support MUST be advertised using discovery and the namespace covered, as explained in Discovering Support below.
- It may be hard to impossible for an implementation to be compliant with features specified at Paging Forwards Through a Result Set in Result Set Management (XEP-0059) [7]. Notably for some order fields, it may be really difficult to not return duplicate items or to no omit items from pages. People interacting with this XEP must be aware of that, and services implementing this XEP SHOULD try to comply with those features, but MAY not if proven too difficult (those features are not required in RSM anyway as the term *MAY* is used).

6 Discovering Support

If a server supports the "order by" protocol, it MUST advertize it including the "urn:xmpp:order-by:1" discovery feature (see Protocol Namespaces regarding issuance of one or more permanent namespaces) in response to a Service Discovery (XEP-0030) [8] information request.In addition to the general feature support, an entity MUST indicated on which protocols Order-By can be used, by using the notation urn:xmpp:order-by:1@*other_namespace*, i.e. a concatenation of:

- this XEP namespace: **urn:xmpp:order-by:1**
- @
- namespace where Order-By is applied

So if Order-By is implemented for Publish-Subscribe (XEP-0060) [9], the service MUST advertise urn:xmpp:order-by:1@http://jabber.org/protocol/pubsub. If Order-By is implemented for Message Archive Management (XEP-0313) [10], it is urn:xmpp:order-by:1@urn:xmpp:mam:2. In the following example, the server example.org advertizes Order-By support, and indicates that it is implemented for Pubsub and MAM:

[6] XEP-0277: Microblogging over XMPP <https://xmpp.org/extensions/xep-0277.html>.
[7] XEP-0059: Result Set Management <https://xmpp.org/extensions/xep-0059.html>.
[8] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[9] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[10] XEP-0313: Message Archive Management <https://xmpp.org/extensions/xep-0313.html>.

Listing 15: 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 16: 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:order-by:1'/>
    <feature var='urn:xmpp:order-by:1@http://jabber.org/protocol/
       pubsub'/>
    <feature var='urn:xmpp:order-by:1@urn:xmpp:mam:2'/>…

  </query>
</iq>
```

7 Business Rules

Several ordering elements may be used, this allows to solve next levels of ordering in case of equality. In this case, the first ordering (i.e. the top most <order> element) is the main one, the second <order> element is used in case of equality, then the next one if a new equality happens and so on.
In case of equality, if no new <order> element is specified, the item order is not guaranteed and is up to the implementation (the implementation MUST keep this order consistent across requests though).

8 Implementation Notes

It may be difficult to find a correct value for <first> and <last> elements of RSM. Indeed, internal ID of items can't be suited for all orderings. For Pubsub service using a SQL database as backend, item ID (XMPP or internal) could be used with a window function such as row_number (supported by major database engines such as PostgreSQL, MariaDB/MySQL or SQLite) over the requested ordering. For instance, on a hypothetical table where items are requested by ascending creation then modification dates after the value ABC (which correspond to XMPP item ID in our case), a request similar to this could be used:

Listing 17: SQL Query to Handle <after> value

```
WITH cte_1 AS
 (SELECT items.id AS id, row_number() OVER (ORDER BY created ASC,
    modified ASC) - 1 AS item_index
    FROM items
    WHERE items.node_id = 123)
 SELECT cte_1.item_index, items.id, items.payload
    FROM items JOIN cte_1 ON items.id = cte_1.id
    WHERE cte_1.item_index > (SELECT cte_1.item_index
        FROM cte_1
        WHERE cte_1.id = "ABC")
    ORDER BY cte_1.item_index ASC
    LIMIT 10;
```

In this example, row_number is decreased by 1 to match RSM index (row_number starts at 1 while RSM index starts at 0), thus the item_index column can be used directly to fill RSM metadata. A Common Table Expression has been used for better readability.

9 Security Considerations

This document introduces no additional security considerations above and beyond those defined in the documents on which it depends.

10 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [11].

11 XMPP Registrar Considerations

11.1 Protocol Namespaces

This specification defines the following XML namespace:

- 'urn:xmpp:order-by:1'

[11] 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/>.

11.2 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.

12 XML Schema

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:order-by:1'
    xmlns='urn:xmpp:order-by:1'
    elementFormDefault='qualified'>

  <xs:element name='order' maxOccurs='unbounded'>
      <xs:complexType>
          <xs:attribute name='by' type='xs:string' use='required'/>
      </xs:complexType>
  </xs:element>

</xs:schema>
```

13 Acknowledgements

Thanks to Philipp Hörist, Evgeny xramtsov, Jonas Schäfer¸ and Holger Weiß for their feedback.