XEP-0059: Result Set Management

Ian Paterson
mailto:ian.paterson@clientside.co.uk
xmpp:ian@zoofy.com

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

Valerie Mercier
mailto:valerie.mercier@orange-ftgroup.com
xmpp:vmercier@jabber.com

Jean-Louis Seguineau
mailto:jls@antepo.com
xmpp:jlseguineau@im.antepo.com

2006-09-20
Version 1.0

Status	Type	Short Name
Draft	Standards Track	rsm

This specification defines an XMPP protocol extension that enables an entity to page through and otherwise manage the receipt of large result sets. The protocol can be used in the context of any XMPP protocol that might send large result sets (such as service discovery, multi-user chat, and publish-subscribe). While the requesting entity in such an interaction can explicitly request the use of result set management, an indication that result set management is in use can also be proactively included by the responding entity when returning a limited result set in response to a query.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2018 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the ”Specification”), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an ”AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

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

In Jabber Search (XEP-0055) [1], Service Discovery (XEP-0030) [2], Publish-Subscribe (XEP-0060) [3], Message Archiving (XEP-0136) [4], and probably other future XMPP extensions, it is possible to receive large dynamic result sets in response to information requests (e.g., a user directory search on a common first name or a service discovery items request sent to a Multi-User Chat (XEP-0045) [5] service). This XMPP protocol extension enables the following functionality for use by other XMPP protocols:

1. Limit the number of items returned.
2. Page forwards or backwards through a result set by retrieving the items in smaller subsets.
3. Discover the size of a result set without retrieving the items themselves.
4. Retrieve a page (subset) of items starting at any point in a result set.

2 Use Cases

2.1 Limiting the Number of Items

In order to limit the number of items of a result set to be returned, the requesting entity specifies a request type of ”set” and the maximum size of the desired subset (via the XML character data of the <max/> element):

Listing 1: Requesting a Limit to the Result Set

```
<iq type='set' from='stpeter@jabber.org/roundabout' to='users.jabber.
    org' id='limit1'>
  <query xmlns='jabber:iq:search'>
    <nick>Pete</nick>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <max>10</max>
    </set>
  </query>
</iq>
```

The responding entity then returns the first items of the result set in order. The number of items is limited to the requested size:

[1] XEP-0055: Jabber Search <https://xmpp.org/extensions/xep-0055.html>.
[2] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[3] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[4] XEP-0136: Message Archiving <https://xmpp.org/extensions/xep-0136.html>.
[5] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.

Listing 2: Returning a Limited Result Set

```
<iq type='result' from='users.jabber.org' to='stpeter@jabber.org/
    roundabout' id='limit1'>
  <query xmlns='jabber:iq:search'>
    <item jid='stpeter@jabber.org'>
      <first>Peter</first>
      <last>Saint-Andre</last>
      <nick>Pete</nick>
    </item>
    .
    [8 more items]
    .
    <item jid='peterpan@neverland.lit'>
      <first>Peter</first>
      <last>Pan</last>
      <nick>Pete</nick>
    </item>
  </query>
</iq>
```

2.2 Paging Forwards Through a Result Set

An entity often needs to retrieve a page of items adjacent to a page it has already received. For examples, when retrieving a complete result set in order page by page, or when a user 'scrolls' forwards one page.
The set of items that match a query MAY change over time, even during the time that a requesting entity pages through the result set (e.g., a set of chatrooms, since rooms can be created and destroyed at any time). The paging protocol outlined in this section is designed so that entities MAY provide the following features:

- Each page of the result set is up-to-date at the time it is sent (not just at the time the first page was sent).
- No items will be omitted from pages not yet sent (even if, after earlier pages were sent, some of the items they contained were removed from the set).
- When paging through the list in order, duplicate items are never received.
- The responding entity maintains no state (or a single minimal state for all requesting entities containing the positions of all recently deleted items).
- Rapid calculation of which items should appear on a requested page by responding entity (even for large result sets).

Note: If a responding entity implements dynamic result sets then receiving entities paging through the complete result set should be aware that it may not correspond to the result set

as it existed at any one point in time.
The request for the first page is the same as when Limiting the Number of Items:

Listing 3: Requesting the First Page of a Result Set

```
<iq type='set' from='stpeter@jabber.org/roundabout' to='users.jabber.
    org' id='page1'>
  <query xmlns='jabber:iq:search'>
    <nick>Pete</nick>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <max>10</max>
    </set>
  </query>
</iq>
```

Responding entity support for paging through a result set is optional. If it does support paging (not just Limiting the Number of Items), then in each page it returns, the responding entity MUST include <first/> and <last/> elements that specify the unique ID (UID) for the first and last items in the page. If there is only one item in the page, then the first and last UIDs MUST be the same. If there are no items in the page, then the <first/> and <last/> elements MUST NOT be included.
The responding entity may generate these UIDs in any way, as long as the UIDs are unique in the context of all possible members of the full result set. Each UID MAY be based on part of the content of its associated item, as shown below, or on an internal table index. Another possible method is to serialize the XML of the item and then hash it to generate the UID. Note: The requesting entity MUST treat all UIDs as opaque.
The responding entity SHOULD also include the number of items in the full result set (which MAY be approximate) encapsulated in a <count/> element. The <first/> element SHOULD include an 'index' attribute. This integer specifies the position within the full set (which MAY be approximate) of the first item in the page. If that item is the first in the full set, then the index SHOULD be '0'. If the last item in the page is the last item in the full set, then the value of the <first/> element's 'index' attribute SHOULD be the specified count minus the number of items in the last page.
Note: The <count/> element and 'index' attribute enable important functionality for requesting entities (for example, a scroll-bar user-interface component). They MAY be omitted, but *only* if it would be either impossible or exceptionally resource intensive to calculate reasonably accurate values.

Listing 4: Returning the First Page of a Result Set

```
<iq type='result' from='users.jabber.org' to='stpeter@jabber.org/
    roundabout' id='page1'>
  <query xmlns='jabber:iq:search'>
    <item jid='stpeter@jabber.org'>
      <first>Peter</first>
      <last>Saint-Andre</last>
```

```
      <nick>Pete</nick>
    </item>
    .
    [8 more items]
    .
    <item jid='peterpan@neverland.lit'>
      <first>Peter</first>
      <last>Pan</last>
      <nick>Pete</nick>
    </item>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <first index='0'>stpeter@jabber.org</first>
      <last>peterpan@neverland.lit</last>
      <count>800</count>
    </set>
  </query>
</iq>
```

The requesting entity can then ask for the next page in the result set by including in its request the UID of the *last* item from the previous page (encapsulated in an <after/> element), along with the maximum number of items to return. Note: If no <after/> element is specified, then the UID defaults to "before the first item in the result set" (i.e., effectively an index of negative one).

Listing 5: Requesting the Second Page of a Result Set

```
<iq type='set' from='stpeter@jabber.org/roundabout' to='users.jabber.
    org' id='page2'>
  <query xmlns='jabber:iq:search'>
    <nick>Pete</nick>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <max>10</max>
      <after>peterpan@neverland.lit</after>
    </set>
  </query>
</iq>
```

The *first* item in the page returned by the responding entity MUST be the item that immediately *follows* the item that the requesting entity indicated in the <after/> element:

Listing 6: Returning the Second Page of a Result Set

```
<iq type='result' from='users.jabber.org' to='stpeter@jabber.org/
    roundabout' id='page2'>
  <query xmlns='jabber:iq:search'>
    <item jid='peter@pixyland.org'>
      <first>Peter</first>
      <last>Pan</last>
```

```
      <nick>Pete</nick>
    </item>
    .
    [8 more items]
    .
    <item jid='peter@rabbit.lit'>
      <first>Peter</first>
      <last>Rabbit</last>
      <nick>Pete</nick>
    </item>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <first index='10'>peter@pixyland.org</first>
      <last>peter@rabbit.lit</last>
      <count>800</count>
    </set>
  </query>
</iq>
```

It may sometimes be necessary to return an empty *page* to the requesting entity. For example, with dynamic result sets the responding entity MAY delete some items from the full result set between requests. Another example occurs when the requesting entity specifies "0" for the maximum number items to return (see Getting the Item Count).

Listing 7: Returning an Empty Page

```
<iq type='result' from='users.jabber.org' to='stpeter@jabber.org/
    roundabout' id='page80'>
  <query xmlns='jabber:iq:search'>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <count>790</count>
    </set>
  </query>
</iq>
```

If there are no items whatsoever in the *full* result set, the responding entity MUST return a response that adheres to the definition of the wrapper protocol (e.g., "jabber:iq:search", "http://jabber.org/protocol/disco#items", or "http://jabber.org/protocol/pubsub"). For both XEP-0055 and XEP-0030, that means the responding entity shall return an empty <query/> element; for XEP-0060, that means the responding entity shall return an empty <pubsub/> element; for XEP-0136, that means the responding entity shall return an empty <list/> or <store/> element.

2.3 Paging Backwards Through a Result Set

The requesting entity MAY ask for the previous page in a result set by including in its request the UID of the *first* item from the page that has already been received (encapsulated in a

<before/> element), along with the maximum number of items to return.

Listing 8: Requesting the Previous Page of a Result Set

```
<iq type='set' from='stpeter@jabber.org/roundabout' to='users.jabber.
    org' id='back1'>
  <query xmlns='jabber:iq:search'>
    <nick>Pete</nick>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <max>10</max>
      <before>peter@pixyland.org</before>
    </set>
  </query>
</iq>
```

The *last* item in the page returned by the responding entity MUST be the item that immediately *preceeds* the item that the requesting entity indicated it has already received:

Listing 9: Returning the Previous Page of a Result Set

```
<iq type='result' from='users.jabber.org' to='stpeter@jabber.org/
    roundabout' id='back1'>
  <query xmlns='jabber:iq:search'>
    <item jid='stpeter@jabber.org'>
      <first>Peter</first>
      <last>Saint-Andre</last>
      <nick>Pete</nick>
    </item>
    .
    [8 more items]
    .
    <item jid='peterpan@neverland.lit'>
      <first>Peter</first>
      <last>Pan</last>
      <nick>Pete</nick>
    </item>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <first index='0'>stpeter@jabber.org</first>
      <last>peterpan@neverland.lit</last>
      <count>800</count>
    </set>
  </query>
</iq>
```

2.4 Page Not Found

The responding entity MUST reply with an 'item-not-found' error if *all* the following circumstances apply:

1. The item specified by the requesting entity via the UID in the <after/> or <before/> element no longer exists (it was deleted after the previous page was sent).

2. The UID itself cannot be used to derive directly the next item within the set (e.g. the alphabetical or numerical order of the UIDs do not specify the order of the items).

3. The responding entity does not remember the position of the deleted item within the full list. (Even if the responding entity bothers to remember the position of each deleted item, it will typically be necessary to expire that 'state' after an implementation-specific period of time.)

Listing 10: Returning a Page-Not-Found Error

```
<iq type='error' from='users.jabber.org' to='stpeter@jabber.org/
    roundabout' id='page2'>
  <query xmlns='jabber:iq:search'>
    <nick>Pete</nick>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <max>10</max>
      <after>peterpan@neverland.lit</after>
    </set>
  </query>
  <error type='cancel'>
    <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

2.5 Requesting the Last Page in a Result Set

The requesting entity MAY ask for the last page in a result set by including in its request an empty <before/> element, and the maximum number of items to return.

Listing 11: Requesting the Last Page of a Result Set

```
<iq type='set' from='stpeter@jabber.org/roundabout' to='users.jabber.
    org' id='page1'>
  <query xmlns='jabber:iq:search'>
    <nick>Pete</nick>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <max>10</max>
      <before/>
    </set>
  </query>
```

```
</iq>
```

2.6 Retrieving a Page Out of Order

The requesting entity MAY choose not to retrieve pages from the result set in order. (For example, when its user drags a user-interface slider to a radically new position within a very large result set.)
Only if the UID before the start (or after the end) of a desired result set page is not known, then the requesting entity MAY request the page that *starts* at a particular index within the result set. It does that by including in its request the index of the *first* item to be returned (encapsulated in an <index/> element), as well as the maximum number of items to return.
Note: For reasons mentioned in Paging Forwards Through a Result Set requesting entities SHOULD, where possible, specify pages using a UID instead of an index.
Note: If the responding entity omitted the <count/> element from previous responses for this result set, then the requesting entity SHOULD assume that the responding entity does not support page retrieval by index for this result set (see error below).

Listing 12: Requesting a Result Page by Index

```
<iq type='set' from='stpeter@jabber.org/roundabout' to='users.jabber.
    org' id='index10'>
  <query xmlns='jabber:iq:search'>
    <nick>Pete</nick>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <max>10</max>
      <index>371</index>
    </set>
  </query>
</iq>
```

The responding entity SHOULD derive the first UID from the specified index (the method used MAY be approximate) before returning the requested result set page in the normal way. If the specified index was "0" then the responding entity SHOULD derive the UID that is the first in the full result set.
Note: The 'index' attribute of the <first/> element MUST be the same as the index specified in the request. If the index specified by the requesting entity is greater than or equal to the number of items in the full set then the responding entity MUST return an empty page (see Paging Forwards Through a Result Set).

Listing 13: Returning a Result Page at an Index

```
<iq type='result' from='users.jabber.org' to='stpeter@jabber.org/
    roundabout' id='index10'>
  <query xmlns='jabber:iq:search'>
    <item jid='peter@pixyland.org'>
      <first>Peter</first>
```

```
      <last>Pan</last>
      <nick>Pete</nick>
    </item>
    .
    [8 more items]
    .
    <item jid='peter@rabbit.lit'>
      <first>Peter</first>
      <last>Rabbit</last>
      <nick>Pete</nick>
    </item>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <first index='371'>peter@pixyland.org</first>
      <last>peter@rabbit.lit</last>
      <count>800</count>
    </set>
  </query>
</iq>
```

If it would be either impossible or exceptionally resource intensive for the responding entity to derive the first UID from the specified index with reasonable accuracy then the responding entity MAY return a <feature-not-implemented/> error.

Listing 14: Returning a Feature-not-Implemented Error

```
<iq type='error' from='users.jabber.org' to='stpeter@jabber.org/
    roundabout' id='index10'>
  <query xmlns='jabber:iq:search'>
    <nick>Pete</nick>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <max>10</max>
      <index>371</index>
    </set>
  </query>
  <error type='cancel'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-
        stanzas'/>
  </error>
</iq>
```

2.7 Getting the Item Count

In order to get the item count of a result set without retrieving the items themselves, the requesting entity simply specifies zero for the maximum size of the result set page:

Listing 15: Requesting the Item Count

```
<iq type='set' from='stpeter@jabber.org/roundabout' to='users.jabber.
    org' id='count1'>
  <query xmlns='jabber:iq:search'>
    <nick>Pete</nick>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <max>0</max>
    </set>
  </query>
</iq>
```

The responding entity then returns the item count, which MAY be approximate rather than precise if determining the exact number of items would be resource-intensive:

Listing 16: Returning the Item Count

```
<iq type='result' from='users.jabber.org' to='stpeter@jabber.org/
    roundabout' id='count1'>
  <query xmlns='jabber:iq:search'>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <count>800</count>
    </set>
  </query>
</iq>
```

Note: The <count/> element MAY be omitted, but *only* if it would be either impossible or exceptionally resource intensive to calculate reasonably accurate values.
Note: If there are no items in the *full* result set then the responding entity MUST return a response that adheres to the definition of the wrapper protocol (see Paging Forwards Through a Result Set).

3 Examples

The foregoing examples show the use of result set management in the context of Jabber Search. In the following examples we show the use of this protocol in the context of Service Discovery. XEP-0136 ("Message Archiving") includes more examples. A future version of this document may also include examples from Publish-Subscribe and other XMPP protocol extensions.

Listing 17: Requesting a Limit to the Result Set

```
<iq type='get' from='stpeter@jabber.org/roundabout' to='conference.
    jabber.org' id='ex2'>
  <query xmlns='http://jabber.org/protocol/disco#items'>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <max>20</max>
    </set>
```

```
  </query>
</iq>
```

Listing 18: Returning a Limited Result Set

```
<iq type='result' from='conference.jabber.org' to='stpeter@jabber.org/
    roundabout' id='ex2'>
  <query xmlns='http://jabber.org/protocol/disco#items'>
    <item jid='12@conference.jabber.org'/>
    <item jid='adium@conference.jabber.org'/>
    <item jid='airhitch@conference.jabber.org'/>
    <item jid='alphaville@conference.jabber.org'/>
    <item jid='apache@conference.jabber.org'/>
    <item jid='argia@conference.jabber.org'/>
    <item jid='armagetron@conference.jabber.org'/>
    <item jid='atticroom123@conference.jabber.org'/>
    <item jid='banquise@conference.jabber.org'/>
    <item jid='bar_paradise@conference.jabber.org'/>
    <item jid='beer@conference.jabber.org'/>
    <item jid='blondie@conference.jabber.org'/>
    <item jid='bpnops@conference.jabber.org'/>
    <item jid='brasileiros@conference.jabber.org'/>
    <item jid='bulgaria@conference.jabber.org'/>
    <item jid='cantinalivre@conference.jabber.org'/>
    <item jid='casablanca@conference.jabber.org'/>
    <item jid='chinortpcrew@conference.jabber.org'/>
    <item jid='coffeetalk@conference.jabber.org'/>
    <item jid='council@conference.jabber.org'/>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <first index='0'>acc3594e844c77696f7a7ba9367ae324b6b958ad</first
          >
      <last>4da91d4b330112f683dddaebf93180b1bd25e95f</last>
      <count>150</count>
    </set>
  </query>
</iq>
```

Listing 19: Requesting a Page Beginning After the Last Item Received

```
<iq type='set' from='stpeter@jabber.org/roundabout' to='conference.
    jabber.org' id='ex3'>
  <query xmlns='http://jabber.org/protocol/disco#items'>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <max>20</max>
      <after>4da91d4b330112f683dddaebf93180b1bd25e95f</after>
    </set>
  </query>
</iq>
```

4 Determining Support

In order for a requesting entity to determine if a responding entity supports result set management, it SHOULD send a Service Discovery information request to the responding entity:

Listing 20: Requesting entity queries responding entity regarding protocol support

```
<iq from='stpeter@jabber.org/roundabout'
    to='conference.jabber.org'
    type='get'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 21: Responding entity communicates protocol support

```
<iq from='conference.jabber.org'
    to='stpeter@jabber.org/roundabout'
    type='result'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='http://jabber.org/protocol/rsm'/>
  </query>
</iq>
```

An entity SHOULD NOT include the result set management extensions defined in this document in its requests if it does not have positive knowledge that the responding entity supports the protocol defined herein. If the responding entity does not understand result set management, it MUST ignore such extensions.
Note: Even if a responding entity understands the result set management protocol, its support for result set management in the context of any given "using protocol" is OPTIONAL (e.g., an implementation could support it in the context of the 'jabber:iq:search' namespace but not in the context of the 'http://jabber.org/protocol/disco#items' namespace). Currently the only way for a requesting entity to determine if a responding entity supports result set management in the context of a given "using protocol" is to include result set management extensions in its request. If the responding entity does not include result set management extensions in its response, then the requesting entity SHOULD NOT include such extensions in future requests wrapped by the "using protocol" namespace.

5 Security Considerations

Security considerations are the responsibility of the using ("wrapper") protocol, such as XEP-0030 for the 'http://jabber.org/protocol/disco#items' namespace, XEP-0055 for the 'jabber:iq:search' namespace, and XEP-0136 for the 'http://jabber.org/protocol/archive'

namespace.

6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [6].

7 XMPP Registrar Considerations

7.1 Protocol Namespaces

The XMPP Registrar [7] includes 'http://jabber.org/protocol/rsm' in its registry of protocol namespaces.

8 XML Schema

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='http://jabber.org/protocol/rsm'
    xmlns='http://jabber.org/protocol/rsm'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0059: http://www.xmpp.org/extensions/xep-0059.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='set'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='after' type='xs:string' minOccurs='0'
            maxOccurs='1'/>
```

[6] 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] 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/>.

```
        <xs:element name='before' type='xs:string' minOccurs='0'
            maxOccurs='1'/>
        <xs:element name='count' type='xs:int' minOccurs='0' maxOccurs
            ='1'/>
        <xs:element ref='first' minOccurs='0' maxOccurs='1'/>
        <xs:element name='index' type='xs:int' minOccurs='0' maxOccurs
            ='1'/>
        <xs:element name='last' type='xs:string' minOccurs='0'
            maxOccurs='1'/>
        <xs:element name='max' type='xs:int' minOccurs='0' maxOccurs='
            1'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='first'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:string'>
          <xs:attribute name='index' type='xs:int' use='optional'/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

9 Acknowledgements

Thanks to Olivier Goffart, Jon Perlow, and Andrew Plotkin for their feedback.