XEP-0433: Extended Channel Search

Jonas Schäfer
mailto:jonas@wielicki.name
xmpp:jonas@wielicki.name

2020-02-27
Version 0.1.0

Status	Type	Short Name
Deferred	Standards Track	ECS

This specification provides a standardised protocol to search for public group chats. In contrast to XEP-0030 (Service Discovery), it works across multiple domains and in contrast to XEP-0055 (Jabber Search) it more clearly handles extensibility.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction & Motivation

The XMPP instant messaging ecosystem is a federated one. This leads to many different group chat service providers existing and interesting public group chats being spread out across them. In order to provide users with a way to find public group chats (henceforth called channels) of interest to them, there needs to be a way to execute a cross-domain search based on keywords.
The protocol in this document provides a general and extensible search for channels across different domains and service types (e.g. MUC vs. MIX). It provides meta-information right in the result set, which allows searching entities to skip additional Service Discovery (XEP-0030) [1] queries against the channels themselves.
The protocol is not only useful for cross-domain search, but also as an alternative to using a Service Discovery (XEP-0030) [2] disco#items request followed by many disco#info requests on a group chat service.

2 Requirements

The protocol:

- must work without state on the server side. This is to allow stateless proxies to be used for pseudonymisation or anonymisation.
- must allow searching the list using a free-text keyword-based search.
- must allow future extensions to the search query and the result.
- must allow retrieving the entire data set (although, for clarification, an operator may choose to turn this off).
- must use completely machine-readable and machine-writable data.

3 Glossary

Channel A public group chat hosted on a Group Chat Service. This can either be a Multi-User Chat (XEP-0045) XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>. room, a Mediated Information eXchange (MIX) (XEP-0369) XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>. channel or something else entirely.

Group Chat Service An entity or deployment which offers multi-user chat relay, such as by Multi-User Chat (XEP-0045) XEP-0045: Multi-User Chat

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

<https://xmpp.org/extensions/xep-0045.html>. or Mediated Information eXchange (MIX) (XEP-0369) XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>..

Search Service An entity which offers the service described in this specification.

Searcher An entity which requests information from the Search Service.

4 Use Cases

4.1 Announcing/discovering support

An entity annouces that it supports serving search queries by publishing the urn:xmpp:channel-search:0:search feature via Service Discovery (XEP-0030) [3]:

Listing 1: XEP-0030 disco#info response

```
<iq from='search.service.example' to='client@user.example' id='id1'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <!-{}- ... -{}->
    <feature var='urn:xmpp:channel-search:0:search'/>
    <!-{}- ... -{}->
  </query>
</iq>
```

4.2 Executing a keyword search

To execute a keyword search, the Searcher MAY first request the search form from the Search Service. Alternatively, the Searcher MAY use the form specified in this document with only the fields which must be implemented by the Search Service.
After obtaining the search form, the Searcher completes the form and sends it back to the Search Service. The Search Service replies with a Result Set Management (XEP-0059) [4] paginated list of results.
The search form is a form conforming to Field Standardization for Data Forms (XEP-0068) [5].

4.2.1 Requesting the search form

To request the search form, an entity sends an empty search element qualified by the urn:xmpp:channel-search:0:search namespace:

[3] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[4] XEP-0059: Result Set Management <https://xmpp.org/extensions/xep-0059.html>.
[5] XEP-0068: Field Data Standardization for Data Forms <https://xmpp.org/extensions/xep-0068.html>.

Listing 2: Searcher requests form from the Search Service

```
<iq from='client@user.example' to='search.service.example' id='id2'
    type='get' xml:lang='en'>
  <search xmlns='urn:xmpp:channel-search:0:search'/>
</iq>
```

The Search Service replies with the form as in the following example:

Listing 3: Search Service returns the search form

```
<iq from='search.service.example' to='client@user.example' id='id2'
    type='result' xml:lang='en'>
  <search xmlns='urn:xmpp:channel-search:0:search'>
    <x xmlns='jabber:x:data' type='form'>
      <field type='hidden' var='FORM_TYPE'>
        <value>urn:xmpp:channel-search:0:search-params</value>
      </field>
      <field type='text-single' var='q' label='Search␣for'/>
      <field type='boolean' var='all' label='Return␣all␣entries␣(
          ignore␣search␣terms)'/>
      <field type='boolean' var='sinname' label='Search␣in␣name'>
        <value>true</value>
      </field>
      <field type='boolean' var='sindescription' label='Search␣in␣
          description'>
        <value>true</value>
      </field>
      <field type='boolean' var='sinaddr' label='Search␣in␣address'>
        <value>true</value>
      </field>
      <field type='text-single' var='min_users' label='Minimum␣number␣
          of␣users'>
        <value>1</value>
      </field>
      <field var="types" type="list-multi" label="Service␣types">
        <value>xep-0045</value>
        <option label='XEP-0045␣Multi␣User␣Chats'><value>xep-0045</
            value></option>
        <option label='XEP-0369␣MIX␣channels'><value>xep-0369</value><
            /option>
      </field>
      <field type='list-single' var='key' label='Sort␣results␣by'>
        <value>{urn:xmpp:channel-search:0:order}nusers</value>
        <option label='Number␣of␣online␣users'><value>{
            urn:xmpp:channel-search:0:order}nusers</value></option>
        <option label='Address'><value>{urn:xmpp:channel-
            search:0:order}address</value></option>
      </field>
    </x>
```

```
</search>
</iq>
```

Note: Not all of the fields shown above are mandatory to implement. See Search Form Fields for a list of fields and their implementation status.

4.2.2 Send a search request

To request the result list for a given search query, a Searcher submits a form with the urn:xmpp:channel-search:0:search-params FORM_TYPE. The Searcher MAY include a Result Set Management (XEP-0059) [6] <set/> element inside the <search/> element. In either case, the Search Service may reply with a RSM-paginated result and the Searcher MUST be able to process that.
If a Searcher composes a search request using a search form template obtained by the Search Service, it MAY omit all fields it does not know or where it does not change the value already supplied by the Search Service.

Listing 4: Searcher submits a form to the Search Service

```
<iq from='client@user.example' to='search.service.example' id='id3'
    type='get' xml:lang='en'>
  <search xmlns='urn:xmpp:channel-search:0:search'>
    <set xmlns="http://jabber.org/protocol/rsm">
      <max>5</max>
    </set>
    <x xmlns="jabber:x:data" type="submit">
      <field type='hidden' var='FORM_TYPE'>
        <value>urn:xmpp:channel-search:0:search-params</value>
      </field>
      <field var="q" type="text-single" label="Search␣for">
        <value>xmpp.org</value>
      </field>
      <field var="key" type="list-single" label="Sort␣results␣by">
        <value>{urn:xmpp:channel-search:0:order}address</value>
        <option label='Number␣of␣online␣users'><value>{
            urn:xmpp:channel-search:0:order}nusers</value></option>
        <option label='Address'><value>{urn:xmpp:channel-
            search:0:order}address</value></option>
      </field>
    </x>
  </search>
</iq>
```

The Search Service calculates the result, paginates it according to its own policy (possibly taking into account the pagination request from the client) and returns a single result page in

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

the response IQ.

Listing 5: Searcher submits a form to the Search Service

```
<iq from='search.service.example' to='client@user.example' id='id3'
    type='result' xml:lang='en'>
  <result xmlns='urn:xmpp:channel-search:0:search'>
    <item address='commteam@muc.xmpp.org'>
      <name>commteam</name>
      <nusers>10</nusers>
      <is-open/>
    </item>
    <!-{}- three more items -{}->
    <item address='operators@muc.xmpp.org'>
      <name>XMPP Service Operators</name>
      <description>Discussion venue for operators of federated XMPP
          services</description>
      <nusers>43</nusers>
      <is-open/>
    </item>
    <set xmlns='http://jabber.org/protocol/rsm'>
      <first>opaque-string-1</first>
      <last>opaque-string-2</last>
      <max>5</max>
    </set>
  </result>
</iq>
```

The result items are <item/> elements wrapped in a <result/> element qualified by the urn:xmpp:channel-search:0:search namespace. The schema, along with extension rules, is described in Result Item Format.
To obtain further results, the Searcher re-submits the identical form with an appropriate Result Set Management (XEP-0059) [7] pagination request, using the information provided by the Search Service in the result <set/> element.
If the sort key requested by the Searcher is not supported by the Search Service, the Search Service MUST reply with <feature-not-implemented/> and the <invalid-sort-key> application defined condition and a modify type:

Listing 6: Search Service replies with feature-not-implemented

```
<iq from='search.service.example' to='client@user.example' id='id3'
    type='error' xml:lang='en'>
  <error type='modify'>
    <feature-not-implemented xmlns='urn:ietf:params:xml:ns:xmpp-
        stanzas'/>
    <invalid-sort-key xmlns='urn:xmpp:channel-search:0:error'/>
```

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

```
  </error>
</iq>
```

If the q field was supplied by the Searcher and the contents of the q field did not yield any term suitable for search, the Search Service MUST reply with an <bad-request/> error and the <invalid-search-terms/> application defined condition. The error type MUST be modify.
The server SHOULD include a human-readable description of the constraints for search terms which were not met in the <text/> element of the error.

Listing 7: Search Service replies with the invalid-search-terms error

```
<iq from='search.service.example' to='client@user.example' id='id3'
    type='error' xml:lang='en'>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>
      Search terms must have at least three characters.
    </text>
    <invalid-search-terms xmlns='urn:xmpp:channel-search:0:error'/>
  </error>
</iq>
```

If the Search Service can not or does (by policy) not want to process the request due to excessive amounts of requests (either by the requesting entity, their domain or any other criteria), it MUST reply with an <resource-constraint/> error with type wait.
The application defined error condition <rate-limit/> MUST be included. This error condition has a RECOMMENDED attribute, retry-after, which provides the amount of seconds after which the Searcher MAY retry the request.
The Search Service MAY include a human-readable description of the rate limit and when to retry in the <text/> element.

Listing 8: Search Service replies with a rate limit notification

```
<iq from='search.service.example' to='client@user.example' id='id3'
    type='error' xml:lang='en'>
  <error type='wait'>
    <resource-constraint xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <rate-limit xmlns='urn:xmpp:channel-search:0:error' retry-after='
        30'/>
  </error>
</iq>
```

Note: See also the rate-limiting related business rules for Searcher entities.
If the Search Service can not or does (by policy) not want to allow a Searcher to retrieve the entire database of channels, it MUST reject queries which set the all field to true with an error as follows:

- If the feature is generally disabled: <not-allowed/> with type cancel
- If the feature is not offered to the Searcher based on its identity: <forbidden/> with type auth

In all cases, the application defined condition <full-set-retrieval-rejected/> MUST be included. The Search Service MAY include a human-readable description of the restrictions around full-list retrieval.
For example, if the full set retrieval had been disabled service-wide by configuration, the Search Service would reply with the following error:

Listing 9: Search Service replies with a full-set-retrieval-rejected error

```
<iq from='search.service.example' to='client@user.example' id='id3'
    type='error' xml:lang='en'>
  <error type='cancel'>
    <not-allowed xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>
      Retrieval of the full database is not allowed.
    </text>
    <full-set-retrieval-rejected xmlns='urn:xmpp:channel-
        search:0:error'/>
  </error>
</iq>
```

If the Searcher provides form fields which are conflicting, the Search Service MUST reply with a <bad-request/> error of type modify. In addition, the <conflicting-fields/> application specific condition MUST be included.
Conflicting field values are those which fundamentally cannot be used in the same query in such a way that the definition of their function is still adhered to. For example, q restricts the results by keywords, but all specifies that *all* entries are returned.
The Search Service SHOULD include a human-readable description of the conflicting fields, referencing to the label values of the involved fields.
The <conflicting-fields/> element MAY have one or more <var/> child elements which refer to var values of the submitted fields. At least one of the referenced fields must be changed in order for a follow-up query to succeed.‘
For example, if the Searcher has set all to true and provided a query in q, the Search Service would reply with an error similar to the following:

Listing 10: Search Service replies with a conflicting-fields error

```
<iq from='search.service.example' to='client@user.example' id='id3'
    type='error' xml:lang='en'>
  <error type='cancel'>
    <not-allowed xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>
      Cannot both return all results and search by keywords.
```

```
    </text>
    <conflicting-fields xmlns='urn:xmpp:channel-search:0:error'>
      <var>all</var>
      <var>q</var>
    </conflicting-fields>
  </error>
</iq>
```

If no field which would define a result set and which is understood by the Search Service is present, it MUST reply with a <bad-request/> error of type cancel.
In addition, the <no-search-conditions/> application defined condition MUST be included.

Listing 11: Search Service replies with the no-search-conditions error

```
<iq from='search.service.example' to='client@user.example' id='id3'
    type='error' xml:lang='en'>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <no-search-conditions xmlns='urn:xmpp:channel-search:0:error'/>
  </error>
</iq>
```

An example of this situation would be a form where neither q nor all are given.

5 Business Rules

- When sending the form template, the Search Service MUST include all fields it supports with their respective default values.
- When submitting a form to the Search Service, a Searcher MAY omit all fields it either does not understand or it has left unchanged.
- When submitting a form to the Search Service, a Searcher MAY omit the <option/> elements.
- When receiving a search form, the Search Service MUST ignore fields with a var value it does not understand.
- When executing a keyword search, the service may process the keyword string in implementation-defined ways. This may include interpreting quotes and other ”special” characters, removing keywords which do not fit internal criteria for suitability and others.
- If the Searcher receives a <rate-limit/> error, the behaviour of the Searcher depends on the retry-after attribute:

  - If the retry-after attribute is present, the Searcher MUST NOT send another search request before the amount of seconds indicated in the retry-after attribute have elapsed. There is no guarantee that the request will be accepted at that time.
  - If the retry-after attribute is *not* present, the Searcher should wait for an implementation-defined amount of time and SHOULD back off exponentially on each subsequent <rate-limit/> error.

- If a search request does not yield any results, the Search Service MUST reply with a <result/> without any <item/> children in a type='result' IQ. Specifically, it MUST NOT reply with an <item-not-found/> error.

- If the all field is set to true and the Search Service allows this operation, all results MUST be included in the result set (and then paginated using Result Set Management (XEP-0059) [8]).

6 Implementation Notes

6.1 Search Form Fields

The search form is extensible as per Field Standardization for Data Forms (XEP-0068) [9]. Implementations are free to add fields on both sides of the exchange, as long as they are properly namespaced using Clark Notation.
The following fields are specified by this document:

var	type	Support level	Description
q	text-single	RECOMMENDED	Input for the keyword-based search. Conflicts with all.
all	boolean	OPTIONAL	Return all results, ignoring text search terms. This does not influence the restrictions imposed by the types field. Conflicts with q.
sinaddress	boolean	RECOMMENDED if q is supported	Control whether the keyword search searches in the address of the channel.

[8] XEP-0059: Result Set Management <https://xmpp.org/extensions/xep-0059.html>.
[9] XEP-0068: Field Data Standardization for Data Forms <https://xmpp.org/extensions/xep-0068.html>.

var	type	Support level	Description
sinname	boolean	REQUIRED if q is supported	Control whether the keyword search searches in the name of the channel.
sindescription	boolean	REQUIRED if q is supported	Control whether the keyword search searches in the textual description of the channel.
types	list-multi	RECOMMENDED	Constrain the service types of channels to return. If not supported, the search MUST only cover Multi-User Chat (XEP-0045) XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>. group chats.
key	list-single	REQUIRED	Select how the results are ordered.

The sort keys specified by this document are the following:

Value	Description
{urn:xmpp:channel-search:0:order}address	Order the results by the address of the channel. This ordering mode guarantees that the Searcher gets a duplicate-free view without omissions when paginating.
{urn:xmpp:channel-search:0:order}nusers	Order the results descendingly by the number of users. This mode does not guarantee that all channels in the database are returned, nor does it guarantee that no duplicates occur across multiple pages.

6.1.1 Extensibility of the Search Form Fields

Search Service implementations may offer custom values for the key field, provided Clark Notation is used to namespace the values.

6.2 Result Item Format

The result items are <item/> elements qualified by the urn:xmpp:channel-search:0:search namespace.
Each <item/> element MUST have an address attribute whose value is a proper JID (as per either RFC 6122 [10] or RFC 7622 [11]). It identifies the channel uniquely.
The following child elements of <item/> are defined by this specification. They are all qualified by the same namespace as <item/> itself.

Element name	Content model	Occurences	Description
name	text character data	1	The human-readable name of the channel.
description	text character data	1	The human-readable description of the channel.
language	text character data	1	A valid xml:lang code which indicates the primary language of the channel.
nusers	non-negative integer character data	1	Number of occupants
service-type	enumeration character data	1	The type of the service which hosts the channel. See below for values and semantics.
is-open	boolean character data	1	If set to true, it indicates that the channel can be joined without extra credentials.
anonymity-mode	enumeration character data	1	Anonymity level of participation. See below for values and semantics.

[10] RFC 6122: Extensible Messaging and Presence Protocol (XMPP): Address Format <http://tools.ietf.org/html/rfc6122>.
[11] RFC 7622: Extensible Messaging and Presence Protocol (XMPP): Address Format <http://tools.ietf.org/html/rfc7622>.

Notes:

1. Any child element may be omitted by a Search Service if the data is not available for any or all rooms.
2. The number of occupants may be stale by an undefined amount of time.
3. A service MAY return future versions of those elements alongside with past versions. Entities need to treat elements with the same name, but different namespace, as entirely different elements.

Value	Description
{urn:xmpp:channel-search:0:anonymity}none	The bare JID of the account or the full JID of one or more devices of each occupant is visible to every other occupant.
muc_semianonymous	As specified in Multi-User Chat (XEP-0045) XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.

Value	Description
xep-0045	A Multi-User Chat (XEP-0045) XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>. service.
xep-0369	A Mediated Information eXchange (MIX) (XEP-0369) XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>. service.

If a Search Service would return entries with the same address with different service types, it SHOULD prefer Mediated Information eXchange (MIX) (XEP-0369) [12] over Multi-User Chat (XEP-0045) [13]. Note that a Search Service MUST NOT return service types the client has not asked for.

[12] XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>.
[13] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.

6.2.1 Extensibility

Search Service implementations are free to add custom child elements to <item/> elements. Searcher implementations MUST be prepared to handle any unknown elements in <item/>, for example by ignoring them.
Additional values for the <anonymity-mode/> element may be specified by future extensions. If an implementation encounters an unknown value on this field, it is RECOMMENDED to either treat it as synonymous to {urn:xmpp:channel-search:0:anonymity}none or request the anonymity mode from the address using a protocol appropriate for the channel's service.

7 Security Considerations

When sending a search form with a q field, the Searcher transmits potentially sensitive information to a third party.

8 IANA Considerations

This specification does not require any interaction with the IANA.

9 XMPP Registrar Considerations

This specification should probably create registries for the various fields it defines, as well as register a form type.

10 XML Schema

To be done.

11 Design Considerations

Instead of rolling a custom protocol for the result items, Jabber Search (XEP-0055) [14] could have been used.
While the result format of Jabber Search (XEP-0055) [15] allows for some generality, it does so in a rather restricted way. It is limited by the data formats and types expressable in Data Forms

[14] XEP-0055: Jabber Search <https://xmpp.org/extensions/xep-0055.html>.
[15] XEP-0055: Jabber Search <https://xmpp.org/extensions/xep-0055.html>.

(XEP-0004) [16]. Sturctured data, beyond lists of text and JIDs, can not be represented with Data Forms (XEP-0004) [17] at all. Machine-readable data would also have to be human-readable at the same time to provide a fallback view for human users. Interationalization of such human-readable data in field values is not possible with Data Forms (XEP-0004) [18].
The advantage of entities being able to process unknown fields in a degraded manner is, principally, still present in the current proposal (although with a different kind of degration). Given the complexity of fully and correctly processing Data Forms (XEP-0004) [19] report data, the slim benefits did, in the eyes of the authors, not outweigh the costs.

12 Acknowledgements

The basis for this protocol was developed for the search.jabber.network public group chat search service. It has been cleaned up for publication as a Standards Track XEP by the author and modified to support more use-cases.

[16] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.
[17] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.
[18] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.
[19] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.