XEP-0503: Server-side spaces

Nicolas Cedilnik
mailto:nicoco@nicoco.fr
xmpp:nicoco@nicoco.fr

2025-03-11
Version 0.1.0

Status	Type	Short Name
Experimental	Standards Track	spaces

This document defines an XMPP protocol to cluster several groupchat rooms together.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2024 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 **1**

2 Terminology **1**

3 Discovering support **1**

4 Protocol **2**

4.1 Fetching spaces from a spaces service 2

4.2 Getting the list of rooms in a specific space 3

4.3 Getting information on a specific space 3

4.4 Getting live updates of the space 5

4.5 Room advertises a parent space 6

4.6 Managing a space 7

4.6.1 Creation 7

4.6.2 Update 7

4.6.3 Deletion 7

5 Business rules **7**

6 XMPP Registrar Considerations **8**

6.1 Protocol Namespaces 8

6.2 Spaces Category/Type 8

6.3 Field Standardization 8

7 Security considerations **9**

8 XML Schema **9**

1 Introduction

A single group chat room is not always enough.
In various situations, one wishes to have several rooms clustered together around a common theme. For instance, large open source software projects use different rooms for user support, development, announcements, etc. Other chat networks solved this by allowing (or even, for some of them, forcing) rooms to be children of a parent entity (examples: Slack's *workspaces*, Discord's *servers*, Mattermost's *teams*, WhatsApp's *communities*, Matrix's *spaces*).
This clustering is already possible in practice by using a dedicated MUC Service (Multi-User Chat (XEP-0045) [1]) to group several rooms, but this limits its *spaces* creation to administrator of servers. This specification proposes a mechanism that:

- is groupchat protocol-independent;
- makes it possible to host several *spaces* on a single MUC service.

Since there are many subtle variations over the concept of *spaces*, this specification voluntarily **does not cover** access control, permissions, membership inside a *space* and its children rooms. Similarly, it does not describe how a *space* holding rooms hosted on several groupchat services in the federated XMPP network would work (but it does not forbid it). It aims at being a lowest common denominator for all sort of *spaces* to be built on.

2 Terminology

- A *space* is an adressable entity acting as the "parent" of several related rooms.
- A *spaces service* is an adressable entity that contains the *spaces*, i.e., it is the parent of multiple *spaces*.

3 Discovering support

Support is discoverd via a disco#info request (Service Discovery (XEP-0030) [2]). The *spaces service* MUST reply with an identity of type "spaces" (plural) of the "conference" category, and the urn:xmpp:spaces:0 feature.

Listing 1: Querying the features of a Spaces Service.

```
<iq type='get'
    from='john@northern.songs.example/walrus'
    to='apple.records.example'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
```

[1] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[2] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

```
</iq>
```

Listing 2: Spaces Service responds with its identity and feature.

```
<iq type='result'
    from='apple.records.example'
    to='john@northern.songs.example/walrus'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='conference' type='spaces' name='Spaces entity'
        >
    ...
    <feature var='urn:xmpp:spaces:0'/>
    ...
  </query>
</iq>
```

4 Protocol

4.1 Fetching spaces from a spaces service

Fetching the list of *spaces* hosted on a *spaces service* is done via a disco#items (Service Discovery (XEP-0030) [3]) request directed at the *spaces service*'s JID at the urn:xmpp:spaces:0 node. Using a JID plus a node makes it possible for an entity to be a *spaces service* while having other identities for which the disco#items behaviour on the JID alone is standardized (e.g., a Multi-User Chat (XEP-0045) [4] service).
The *space service* MUST respond with the list of available spaces for the requesting entity (considerations regarding access control and visibility of spaces are outside the scope of this specification). Each space has a JID.

Listing 3: Querying the list of spaces

```
<iq type='get'
    from='george@harrisongs.lsd.example/flowerpower'
    to='apple.records.example'>
  <query xmlns='http://jabber.org/protocol/disco#items'
         node='urn:xmpp:spaces:0'/>
</iq>
```

Listing 4: Space Service responds with the list of spaces it hosts

```
<iq type='result'
    from='apple.records.example'
    to='george@harrisongs.lsd.example/flowerpower'>
  <query xmlns='http://jabber.org/protocol/disco#items'
```

[3] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.
[4] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.

```
        node='urn:xmpp:spaces:0'>
    <item jid='space1@apple.records.example' name='Space #1' />
    <item jid='space2@apple.records.example' name='Space #2' />
    <item jid='space3@apple.records.example' name='Space #3' />
  </query>
</iq>
```

If the *spaces service* hosts a large number of *spaces*, implementations MAY paginate the results using Result Set Management (XEP-0059) [5].

4.2 Getting the list of rooms in a specific space

Fetching the list of rooms that are children of a *space* is done via a disco#items directed at the JID of the space on the space#items node. Using a node on top of the JID makes it possible for a space to be a room itself, which is possible but not required by this specification.
In this case, the *space* JID MUST be present in the list of rooms.

Listing 5: Querying the list of spaces

```
<iq type='get'
    from='paul@northern.songs.example/fool-hill'
    to='space1@apple.records.example'>
  <query xmlns='http://jabber.org/protocol/disco#items'
         node='space#items' />
</iq>
```

Listing 6: Entity responds with the rooms (children) of this space

```
<iq type='result'
    from='space1@apple.records.example'
    to='paul@northern.songs.example/fool-hill'>
  <query xmlns='http://jabber.org/protocol/disco#items'
         node='space#items'>
    <item jid='room1@apple.records.example' name='Room #1' />
    <item jid='room2@apple.records.example' name='Room #2' />
    <item jid='room3@apple.records.example' name='Room #3' />
    <!-{}- If the space JID is also a room JID, then it MUST be inside
        that list -{}->
    <item jid='space1@apple.records.example' name='Lobby' />
  </query>
</iq>
```

4.3 Getting information on a specific space

Getting information on a space is achieved by a disco#info query on the space JID. The *space* responds with an identity of category "conference" and type "space" (singular), along with a

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

urn:xmpp:spaces:0 feature.
The space responds with a Service Discovery Extensions (XEP-0128) [6] element of FORM_TYPE=space#info.

Listing 7: Querying a specific space

```
<iq type='get'
    from='paul@northern.songs.example/fool-hill'
    to='space1@apple.records.example'>
  <query xmlns='http://jabber.org/protocol/disco#info' />
</iq>
```

Listing 8: Entity responds

```
<iq type='result'
    from='space1@apple.records.example'
    to='paul@northern.songs.example/fool-hill'>
  <query xmlns='http://jabber.org/protocol/disco#info' '>
␣␣␣␣<identity␣category='conference'␣type='space'␣name='Space #1'>
␣␣␣␣...
␣␣␣␣<feature␣var='urn:xmpp:spaces:0'␣/>
␣␣␣␣<x␣xmlns='jabber:x:data'␣type='result'>
␣␣␣␣␣␣<field␣var='FORM_TYPE'␣type='hidden'>
␣␣␣␣␣␣␣␣<value>space#info</value>
␣␣␣␣␣␣</field>
␣␣␣␣␣␣<field␣var='space#jid'>
␣␣␣␣␣␣␣␣<value>space1@apple.records.example</value>
␣␣␣␣␣␣</field>
␣␣␣␣␣␣<field␣var='space#name'>
␣␣␣␣␣␣␣␣<value>Space␣#1</value>
␣␣␣␣␣␣</field>
␣␣␣␣␣␣<field␣var='space#desc'>
␣␣␣␣␣␣␣␣<value>Here␣we␣discuss␣stuff␣of␣the␣utmost␣importance.</value>
␣␣␣␣␣␣</field>
␣␣␣␣␣␣<field␣var='space#avatar'>
␣␣␣␣␣␣␣␣<value>https://apple.records.example/logo-highres.jpg</value>
␣␣␣␣␣␣␣␣<value>https://apple.records.example/logo-lowres.jpg</value>
␣␣␣␣␣␣</field>
␣␣␣␣␣␣<field␣var='space#avatarhash'>
␣␣␣␣␣␣␣␣<value>XXX</value>
␣␣␣␣␣␣␣␣<value>YYY</value>
␣␣␣␣␣␣</field>
␣␣␣␣␣␣<field␣var='space#rooms'>
␣␣␣␣␣␣␣␣<value>room1@apple.records.example</value>
␣␣␣␣␣␣␣␣<value>room2@apple.records.example</value>
␣␣␣␣␣␣␣␣<value>room3@apple.records.example</value>
␣␣␣␣␣␣</field>
```

[6] XEP-0128: Service Discovery Extensions <https://xmpp.org/extensions/xep-0128.html>.

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

4.4 Getting live updates of the space

A *spaces service* SHOULD also implement a minimal subset of features of a Pubsub Service (Publish-Subscribe (XEP-0060) [7]) for other entities to subscribe to updates of the informations of a given space, such as room additions/removal, or name/avatar/description changes, without needing to poll regularly.
To receive live updates on a given space, an entity sends a subscription request on the JID of the *space* directed at the urn:xmpp:spaces:0 node. *Spaces services* MAY automatically subscribe entities that join a room that is a children of a given space.

Listing 9: Subscribing to live of updates of a space

```
<iq type="set"
    from="george@martin.com.example/outerspace"
    to="space1@apple.records.example">
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscribe node='urn:xmpp:spaces:0' jid='george@martin.com.example
        ' />
  </pubsub>
</iq>
<iq type="result"
    to="space1@apple.records.example"
    from="george@martin.com.example/outerspace">
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscription node='urn:xmpp:spaces:0' subscription='subscribed' /
        >
  </pubsub>
</iq>
```

Listing 10: Space sends an update of the space

```
<message from='space1@apple.records.example'
         to='francisco@denmark.lit.example'>
  <event xmlns='http://jabber.org/protocol/pubsub#event'>
    <items node='urn:xmpp:spaces:0'>
      <item id='latest'>
        <x xmlns='jabber:x:data' type='result'>
          <field var='FORM_TYPE' type='hidden'>
            <value>space#info</value>
          </field>
          <field var='space#jid'>
            <value>space1@apple.records.example</value>
```

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

```
          </field>
          <field var='space#name'>
            <value>We will now call this one Spaces #3 just to make
                things confusing.</value>
          </field>
          ...
          <field var='space#rooms'>
            <value>room1@apple.records.example</value>
            <value>room2@apple.records.example</value>
            <value>room3@apple.records.example</value>
            <value>room4@apple.records.example</value>
          </field>
        </x>
      </item>
    </items>
  </event>
</message>
```

Alternatively, a *space service* MAY send updates in the form of headline messages containing the space#info form, emanating from the *space* JID.

4.5 Room advertises a parent space

If a room is part of a *space*, it MUST return the space#info form as part of its Service Discovery Extensions (XEP-0128) [8] reponse, and advertise the urn:xmpp:spaces:0 feature.

Listing 11: Querying room info

```
<iq type='get'
    from='ringo@drums.boom.example/kick'
    to='room1@apple.records.example'>
  <query xmlns='http://jabber.org/protocol/disco#info' />
</iq>
```

Listing 12: Room responds

```
<iq type='get'
    from='room1@apple.records.example'
    to='ringo@drums.boom.example/kick'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category="conference" type="text" />
    ...
    <feature var='urn:xmpp:spaces:0' />
    <x xmlns='jabber:x:data' type='result'>
      <field var='FORM_TYPE' type='hidden'>
        <value>space#info</value>
```

[8] XEP-0128: Service Discovery Extensions <https://xmpp.org/extensions/xep-0128.html>.

```
      </field>
      <field var='space#jid'>
        <value>space1@apple.records.example</value>
      </field>
      <field var='space#name'>
        <value>Space #1</value>
      </field>
      ...
    </x>
  </query>
</iq>
```

4.6 Managing a space

The *spaces service* SHOULD implement Ad-Hoc Commands (XEP-0050) [9] for entities to create, update and delete adhoc commands. Creating and updating a *space* rely on the minimal space#info form defined in this document.

4.6.1 Creation

Creating a *space* uses a command on the *spaces service* JID directed at the space#create node. The entity responds with the space#info form.

4.6.2 Update

Updating a *space* uses a command on the *space* JID directed at the space#update node. The entity responds with the space#info form.

4.6.3 Deletion

Deleting a *space* uses a command on the *space* JID directed at the space#delete node. Whether or not the rooms in this space shall be deleted on is out of scope of this specification.

5 Business rules

A *spaces service* can be a dedicated component, but this is **not a requirement**. It can also be a MUC service (Multi-User Chat (XEP-0045) [10]) if it hosts rooms too. It MUST also be a pubsub service if it broadcast updates via Publish-Subscribe (XEP-0060) [11].

[9] XEP-0050: Ad-Hoc Commands <https://xmpp.org/extensions/xep-0050.html>.
[10] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[11] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

This is **not a requirement**, but a *space* can be a room itself. In this case, this room can act as a "lobby" (general purpose room) for this *space*. Permissions and roles of this room can propagate to the rooms of this space, and/or act as the permission model for updating the *space*.
A room MAY be part of different *spaces*. In this case, it MUST advertise multiple space#info forms in its disco#info (Service Discovery Extensions (XEP-0128) [12]).

6 XMPP Registrar Considerations

6.1 Protocol Namespaces

New namespace "urn:xmpp:spaces:0"

6.2 Spaces Category/Type

New category "spaces" for "conference" identity. (spaces service)
New category "space" for "conference" identity. (a specific space)

6.3 Field Standardization

This document defines a new FORM_TYPE: space#info.

```
<form_type>
  <name>space#info</name>
  <doc>XEP-xxxx</doc>
  <desc>
    Description of a space, meant to be used in disco#info XEP-0128
  </desc>
  <field
     var='space#jid'
     type='text'
     label='JID␣this␣space.'/>
  <field
     var='space#name'
     type='text'
     label='Human-readable␣name␣of␣the␣space' />
  <field
     var='space#desc'
     type='text'
     label='Human-readable␣description␣of␣the␣purpose␣of␣this␣space'/>
  <field
```

[12] XEP-0128: Service Discovery Extensions <https://xmpp.org/extensions/xep-0128.html>.

```
      var='space#avatarhash'
      type='text-multi'
      label='Hashes of the avatar representing the space'/>
  <field
      var='space#avatar'
      type='text-multi'
      label='Sources for the avatar'/>
  <field
      var='space#rooms'
      type='jid-multi'
      label='Rooms that are part of this space'/>
</form_type>
```

7 Security considerations

Security considerations are related to access control, and are out of scope of this document.

8 XML Schema

No new schema is defined in this document.