XEP-0421: Occupant identifiers for semi-anonymous MUCs

Marvin Wißfeld
mailto:xmpp@larma.de
xmpp:jabber@larma.de

2025-04-09
Version 1.0.1

Status	Type	Short Name
Draft	Standards Track	occupant-id

This specification defines a method that allows clients to identify a MUC participant across reconnects and renames. It thus prevents impersonification of semi-anonymous users.

Contents

1 Introduction

Multi-User Chat (XEP-0045) [1] allows the creation of semi-anonymous multi-user text chats where the real JID of a occupant can not be discovered by other MUC occupants except moderators. As such users can freely join with using any identity of their choice, allowing to impersonate users while they are not online.
With recent standard extensions, it becomes more relevant to be able to know if the occupant that sends one message is the same as the sender of another message, for example for Last Message Correction (XEP-0308) [2]. At the same time it becomes harder for clients to determine this, for example due to the use of Message Archive Management (XEP-0313) [3] with MUCs.
This specification defines a method to combat issues arising out of the pseudonymity of MUC occupants while at the same time ensuring their privacy by not revealing their real JID to other occupants.

2 Discovering support

If a MUC room implements occupant identifiers, it MUST specify the 'urn:xmpp:occupant-id:0' feature in its service discovery information features as specified in Service Discovery (XEP-0030) [4].

Listing 1: Client requests information about a MUC

```
<iq type='get'
    to='coven@chat.shakespeare.lit'
    from='hag66@shakespeare.lit/pda'
    id='info1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 2: MUC advertises support for occupant identifiers

```
<iq type='result'
    to='hag66@shakespeare.lit/pda'
    from='coven@chat.shakespeare.lit'
    id='info1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
...
    <feature var='urn:xmpp:occupant-id:0'/>
...
  </query>
</iq>
```

[1]XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[2]XEP-0308: Last Message Correction <https://xmpp.org/extensions/xep-0308.html>.
[3]XEP-0313: Message Archive Management <https://xmpp.org/extensions/xep-0313.html>.
[4]XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

3 Use Cases

3.1 Entering a Room

When a user enters a room, they send a presence to claim the nickname in the MUC. A MUC that supports occupant identifiers attaches an <occupant-id> element within the "urn:xmpp:occupant-id:0" namespace to the presence sent to all occupants in the room.

Listing 3: Client joins a room

```
<presence
    from='hag66@shakespeare.lit/pda'
    id='n13mt3l'
    to='coven@chat.shakespeare.lit/thirdwitch'>
  <x xmlns='http://jabber.org/protocol/muc'/>
</presence>
```

Listing 4: Service sends new occupant's presence to all occupants

```
<presence
    from='coven@chat.shakespeare.lit/thirdwitch'
    id='27C55F89-1C6A-459A-9EB5-77690145D624'
    to='crone1@shakespeare.lit/desktop'>
  <x xmlns='http://jabber.org/protocol/muc#user' />
  <occupant-id xmlns="urn:xmpp:occupant-id:0" id="
      dd72603deec90a38ba552f7c68cbcc61bca202cd" />
</presence>
```

3.2 Sending a Message to All Occupants

An occupant sends a message to all other occupants in the room by sending a message of type "groupchat" to the <room@service>. A MUC supporting occupant identifiers attaches an <occupant-id> element within the "urn:xmpp:occupant-id:0" to the message sent to all occupants in the room.

Listing 5: Client sends a message to all occupants

```
<message
    from='hag66@shakespeare.lit/pda'
    id='hysf1v37'
    to='coven@chat.shakespeare.lit'
    type='groupchat'>
  <body>Harpier cries: 'tis␣time,␣'tis time.</body>
</message>
```

Listing 6: Service reflects message to all occupants

```
<message
    from='coven@chat.shakespeare.lit/thirdwitch'
    id='hysf1v37'
    to='crone1@shakespeare.lit/desktop'
    type='groupchat'>
  <body>Harpier cries: 'tis␣time,␣'tis time.</body>
  <occupant-id xmlns="urn:xmpp:occupant-id:0" id="
     dd72603deec90a38ba552f7c68cbcc61bca202cd" />
</message>
```

4 Business Rules

Messages and presences MUST NOT contain more than one <occupant-id> element. If the message or presence received by the MUC service already contains <occupant-id> element, the MUC service MUST replace such element before reflecting the message or presence including it.
The <occupant-id> element MUST be attached to every message and every presence sent by a MUC. This includes messages sent as part of the discussion history after joining a room, requested via Message Archive Management (XEP-0313) [5] or any other means.
The <occupant-id> element MUST be ignored if support for the feature is not announced via Service Discovery (XEP-0030) [6], as malicious clients might forge occupant identifiers if the room does not support them.
A MUC service MAY allow the administrator to enable or disable occupant identifiers on a per-room basis. If occupant identifiers are force enabled for all rooms on a MUC service, it SHOULD additionally specify the 'urn:xmpp:occupant-id:0' feature on the MUC service. It MUST NOT specify the feature on the service otherwise.

4.1 Occupant ID generation

The occupant identifier MUST be generated such that it is stable. This means that if a user joins the same room a second time, the occupant identifier MUST be the same as was assigned the first time. A user in the sense of this specification is identified by its real bare JID.
The occupant identifier MUST be generated such that it is unique. This means that it MUST be sufficiently improbable that one user is able to re-create the occupant identifier of another user.
The occupant identifier MUST be generated such that it is pseudonymous. This means that it MUST be sufficiently hard to determine the real bare JID of an occupant from its occupant identifier. Additionally, a MUC service SHOULD generate the identifier such that the occupant identifier of a user in one room of the service does not match the occupant identifier of the same user in another room of the same service. If the MUC service generates the same

[5]XEP-0313: Message Archive Management <https://xmpp.org/extensions/xep-0313.html>.
[6]XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

occupant identifier for the same user in different rooms, information shared using different nicknames and in different rooms could be combined through the occupant identifier and thereby unintentionally reveal information about the user. To guarantee the pseudonymity property, the server MUST NOT generate an occupant identifier by only hashing the real bare JID using static, guessable or discoverable parameters.
The occupant identifier MUST have a maximum length of 128 characters. The recipient MUST consider the occupant identifier to be an opaque string.
One way to ensure these properties is to generate a private secret key for every room and use an HMAC algorithm with a sufficiently secure hash function to generate the occupant identifier from the real bare JID and that secret key. Alternatively, the service can generate a single private secret key for the whole service and use an HMAC algorithm to generate the occupant identifier from the real bare JID, the room bare JID and the service secret key. This procedure ensures all the required properties with minimal server side storage requirements.

Listing 7: Pseudo-code of occupant identifier generation

```
room_secret_key := RANDOM_BYTES(32) # Stored with room state
occupant-id := HEX(HMAC_SHA1(room_secret_key, user_bare_jid))

# or #

service_secret_key := RANDOM_BYTES(32) # Stored with service state
occupant-id := HEX(HMAC_SHA1(service_secret_key, room_bare_jid || NUL
   || user_bare_jid))
```

5 Security Considerations

If a MUC uses occupant identifiers, nickname changes will be visible to all occupants of the room. Clients may warn users about this circumstance before joining the room or when changing the nickname.
When the MUC service does not support this specification, the server will likely forward any <occupant-id> included in <message>s sent by other room occupants and reflected by the MUC service. Receiving clients must be careful to only process occupant identifiers if the MUC server advertises support for this specification as described in the Discovering support section.
The pseudonymity property of occupant identifiers is crucial to not accidentally reveal an occupant's real bare JID to other room occupants. Specifically, a simple hash over the occupant's real bare JID is not sufficient as an occupant identifier, as unsalted hashes can be reversed easily based on a dictionary of candidate JIDs. Review the Occupant ID generation section for more details.

6 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [7].

7 XMPP Registrar Considerations

7.1 Protocol Namespaces

The XMPP Registrar [8] includes 'urn:xmpp:occupant-id:0' in its registry of protocol namespaces (see <https://xmpp.org/registrar/namespaces.html>).

- urn:xmpp:occupant-id:0

8 XML Schema

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="urn:xmpp:occupant-id:0"
           xmlns="urn:xmpp:occupant-id:0"
           elementFormDefault="qualified">

    <xs:element name="occupant-id">
      <xs:complexType>
        <xs:attribute name="id" type="OccupantIdentifier" use="
            required" />
      </xs:complexType>
    </xs:element>

    <xs:simpleType name="OccupantIdentifier">
        <xs:restriction base="xs:string">
            <xs:minLength value="1"/>
            <xs:maxLength value="128"/>
        </xs:restriction>
    </xs:simpleType>

</xs:schema>
```

[7] 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/>.

[8] 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/>.