



# XMPP

## XEP-0289: Federated MUC for Constrained Environments

Kevin Smith

<mailto:kevin.smith@isode.com>

<xmpp:kevin.smith@isode.com>

2010-11-29

Version 0.1

Status	Type	Short Name
Deferred	Standards Track	FMUC

This document provides a protocol for reducing the bandwidth cost of local users contributing to a remote MUC over a constrained link through local proxying of the MUC room.

# Legal

## Copyright

This XMPP Extension Protocol is copyright © 1999 - 2012 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 <http://xmpp.org/about-xmpp/xsf/xsf-ipr-policy/> or obtained by writing to XMPP Standards Foundation, 1899 Wynkoop Street, Suite 600, Denver, CO 80202 USA).

# Contents

1	Introduction	1
2	Requirements	1
3	Addressing	1
4	Actors	1
5	Use Cases	2
5.1	Joining	2
5.1.1	Success case	2
5.1.2	Failure case	3
5.1.3	Joining the MUC directly	4
5.2	Parting	5
5.2.1	Proxy-connected Users	5
5.2.2	Direct-connection Users	6
5.2.3	Status changes	6
5.3	Sending a Message to All Occupants	6
5.3.1	Normal use	6
5.3.2	Fire and Forget	7
5.4	Administration Use Cases	8
6	Business Rules	8
7	Security Considerations	9
8	IANA Considerations	9
9	XMPP Registrar Considerations	9
10	XML Schema	9

## 1 Introduction

MUC uses lots of bandwidth. Sometimes server to server traffic is heavily constrained. This limits the amount of traffic going across s2s through local proxying for remote MUC rooms. It requires no setup in advance, and needs no bandwidth for remote rooms without local occupants. The premise is that a proxy room joins another room, and receives stanzas from the MUC just as another occupant would.

## 2 Requirements

If appropriately configured, avoid bandwidth use that isn't strictly necessary for message exchange.

Allow an endpoint to scale gracefully up to the usual full MUC chat service as bandwidth allows.

## 3 Addressing

Each local representation has a different address for the federated MUC so that standard XMPP routing rules can be used, and servers do not need to be modified. To generate the JID through which a user can join a federated MUC, the joining client should apply [JID Escaping](#)<sup>1</sup> to the JID of the MUC, and use this as the node part of a JID with the host of the mirroring domain. For example, if a client is connected to the server 'remote.example.com', which has a mirroring service 'mirror.remote.example.com', and the user wants to join the MUC 'jabberchat@talk.example.com', their client would generate a federated MUC JID of jabberchat\40talk.example.com@mirror.remote.example.com for them to use.

## 4 Actors

The following JIDs are used in this document.

- example.com - service
- talk.example.com - MUC service on example.com.
- curtis@example.com - User on example.com
- jabberchat@talk.example.com - MUC room.
- remote.example.com - remote service, connected to example.com over constrained link

---

<sup>1</sup>XEP-0106: JID Escaping <<http://xmpp.org/extensions/xep-0106.html>>.

- kev@remote.example.com/Swift - user connected to a remote service
- mirror.remote.example.com - MUC mirroring service on remote.example.com
- jabberchat\40talk.example.com@mirror.remote.example.com - slave room.

## 5 Use Cases

### 5.1 Joining

#### 5.1.1 Success case

kev@remote.example.com/Swift joining jabberchat@talk.example.com through a pre-known mirror.remote.example.com service. At this point mirror.remote.example.com knows nothing of the jabberchat@talk.example.com MUC, and no existing proxying is in place beyond mirror.remote.example.com being willing to proxy for kev@remote.example.com

Listing 1: User joins MUC through a proxy

```
<presence
  from='kev@remote.example.com/Swift'
  to='jabberchat\40talk.example.com@mirror.remote.example.com/Kev'>
  <x xmlns='http://jabber.org/protocol/muc' />
</presence>
```

mirror.example.com then un-escapes 'jabberchat\40talk.example.com', and joins jabberchat@talk.example.com (the master), saying it's a room mirror.

Listing 2: Proxy service joins the target MUC

```
<presence
  from='jabberchat\40talk.example.com@mirror.remote.example.com'
  to='jabberchat@talk.example.com/Kev'>
  <fmuc xmlns='http://isode.com/protocol/fmuc' from='kev@remote.
    example.com/Swift' />
</presence>
```

jabberchat@talk.example.com recognises that the mirror service is now mirroring it, and performs the usual ACL checks as if kev@example.com/Swift had joined directly, sending presence to all occupants as normal. For all in-room routing, the slave is now treated as an occupant, and the slave is expected to do fan-out to its users as it is itself a MUC.

Listing 3: MUC confirms room join

```
<presence
  from='jabberchat@talk.example.com/Kev'
  to='jabberchat\40talk.example.com@mirror.remote.example.com'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
```

```

    <item affiliation='none' role='participant' />
  </x>
</presence>

```

The slave then fans-out.

Listing 4: Proxy delivers the join to local users

```

<presence
  from='jabberchat\40talk.example.com@mirror.remote.example.com/Kev'
  to='kev@remote.example.com/Swift'
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='none' role='participant' />
  </x>
</presence>

```

### 5.1.2 Failure case

If the master doesn't allow the user to join, they send the standard MUC error to the slave. Note that for stanzas sent to a user on the slave (such as this join error), it sends to the full MUC JID of the user on the slave, not to the slave room as it does with most other stanzas.

Listing 5: Master rejects joins

```

<presence
  from='jabberchat@talk.example.com'
  to='jabberchat\40talk.example.com@mirror.remote.example.com/Kev'
  type='error'>
  <x xmlns='http://jabber.org/protocol/muc' />
  <error type='auth'>
    <registration-required xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'
      />
  </error>
</presence>

```

The proxy then delivers this to the user

Listing 6: Proxy delivers the join failure to the user

```

<presence
  from='jabberchat\40talk.example.com@mirror.remote.example.com'
  to='kev@remote.example.com/Swift'
  type='error'>
  <x xmlns='http://jabber.org/protocol/muc' />
  <error type='auth'>
    <registration-required xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'
      />
  </error>
</presence>

```

```
</error>
</presence>
```

### 5.1.3 Joining the MUC directly

Now when a user joins the master directly it will do usual presence distribution to occupants (remembering the slave is an occupant). Status codes are omitted from this example, see [Multi-User Chat](#)<sup>2</sup> for those.

Listing 7: User joins the master MUC directly

```
<presence
  from='curtis@example.com/Swift'
  to='jabberchat@talk.example.com/Curtis'>
  <x xmlns='http://jabber.org/protocol/muc' />
</presence>
```

Listing 8: MUC delivers the join to its occupants

```
<presence
  from='jabberchat@talk.example.com/Curtis'
  to='curtis@example.com/Swift'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='owner' role='moderator' />
  </x>
</presence>

<presence
  from='jabberchat@talk.example.com/Curtis'
  to='jabberchat\@talk.example.com@mirror.remote.example.com'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='admin' role='moderator' />
  </x>
</presence>
```

Listing 9: Proxy delivers join to its occupants

```
<presence
  from='jabberchat\@talk.example.com@mirror.remote.example.com/
    Curtis'
  to='kev@remote.example.com/Swift'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='owner' role='moderator' />
  </x>
</presence>
```

<sup>2</sup>XEP-0045: Multi-User Chat <<http://xmpp.org/extensions/xep-0045.html>>.

## 5.2 Parting

### 5.2.1 Proxy-connected Users

The flow for a user leaving the proxy room is much the same as joining the proxy room:

Listing 10: User leaves the proxy room

```
<presence
  from='kev@remote.example.com/Swift'
  to='jabberchat\40talk.example.com@mirror.remote.example.com/Kev'
  type='unavailable' />
```

Listing 11: Proxy sends part to the MUC

```
<presence
  from='jabberchat\40talk.example.com@mirror.remote.example.com'
  to='jabberchat@talk.example.com/Kev'
  type='unavailable' />
```

Listing 12: MUC transmits part to occupants

```
<presence
  from='jabberchat@talk.example.com/Kev'
  to='jabberchat\40talk.example.com@mirror.remote.example.com'
  type='unavailable'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='none' role='none' />
  </x>
</presence>

<presence
  from='jabberchat@talk.example.com/Kev'
  to='curtis@example.com/Swift'
  type='unavailable'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='none' role='none' />
  </x>
</presence>
```

Listing 13: Proxy sends part to local users

```
<presence
  from='jabberchat\40talk.example.com@mirror.remote.example.com/Kev'
  to='kev@remote.example.com/Swift' {}'
  type='unavailable'>
  <x xmlns='http://jabber.org/protocol/muc#user'>
    <item affiliation='none' role='none' />
    <status_code='110' />
  </x>
</presence>
```

When the master MUC receives a parting presence from the only user of the proxy, the proxy itself also leaves the room. This means that as long as no users of the proxy are in the room, it is causing no traffic on the s2s link.

### 5.2.2 Direct-connection Users

Distribution of presence for users parting when connected directly to the MUC is identical to distribution of presence for users joining directly to the MUC.

### 5.2.3 Status changes

Distribution of presence for users changing status is the same as that for joining and parting.

## 5.3 Sending a Message to All Occupants

### 5.3.1 Normal use

Normal fan-out like presence

Listing 14: Proxy user sends a message to the room

```
<message
  from='kev@remote.example.com/Swift'
  to='jabberchat\40talk.example.com@mirror.remote.example.com'
  type='groupchat'>
  <body>[[Unclassified]] It's_getting_warm_in_here.</body>
</message>
```

Listing 15: Proxy sends the message to the MUC

```
<message
  from='jabberchat\40talk.example.com@mirror.remote.example.com/Kev'
  to='jabberchat@talk.example.com'
  type='groupchat'>
  <body>[[Unclassified]] It's_getting_warm_in_here.</body>
</message>
```

If the proxy is not using fire and forget mode (see below), it MUST NOT fan out this message to local users until it receives the message copy from the MUC.

Listing 16: MUC sends the message to the occupants

```
<message
  from='jabberchat@talk.example.com'
  to='jabberchat\40talk.example.com@mirror.remote.example.com'
```

```

    type='groupchat'>
    <body>[[Unclassified]] It's getting warm in here.</body>
</message>

<message
  from='jabberchat@talk.example.com'
  to='curtis@example.com/Swift'
  type='groupchat'>
  <body>[[Unclassified]] It's getting warm in here.</body>
</message>

```

When receiving the message copy, the proxy MUST then distribute to proxied occupants.

Listing 17: Proxy sends the message to the proxied users

```

<message
  from='jabberchat\@talk.example.com@mirror.remote.example.com/Kev'
  to='kev@remote.example.com/Swift'
  type='groupchat'>
  <body>[[Unclassified]] It's getting warm in here.</body>
</message>

```

### 5.3.2 Fire and Forget

When dealing with very constrained s2s links, the extra round-trip involved with the MUC sending the message back to the proxy may be unacceptable. In this case, the proxy MAY include the <nomirror> element. If the MUC receives a message from a proxy with <nomirror>, it MUST NOT resend this message to the proxy during its usual fan-out, but MUST send it to other occupants as usual. If sending a message with <nomirror>, the proxy MUST perform fan-out as if the MUC had sent the message back to it.

Note that this use introduces unfortunate side-effects, such as messages appearing out of order, depending on whether connected directly to the MUC, or through a proxy. Also, messages rejected by the MUC may already have been delivered to users on a proxy. As such, a proxy SHOULD only use <nomirror> in environments where these side-effects are understood.

Listing 18: Proxy user sends a message to the room

```

<message
  from='kev@remote.example.com/Swift'
  to='jabberchat\@talk.example.com@mirror.remote.example.com'
  type='groupchat'>
  <body>[[Unclassified]] It's getting warm in here.</body>
</message>

```

Listing 19: Proxy sends the message to the MUC

```

<message

```

```

    from='jabberchat\40talk.example.com@mirror.remote.example.com/Kev'
    to='jabberchat@talk.example.com'
    type='groupchat'>
    <body>[[Unclassified]] It's getting warm in here.</body>
</nomirror xmlns='http://isode.com/protocol/fmuc' />
</message>

```

If the proxy is using fire and forget mode, it MUST fan out this message to local users now, instead of waiting until it receives the message copy from the MUC.

Listing 20: Proxy sends the message to the proxied users

```

<message
  from='jabberchat\40talk.example.com@mirror.remote.example.com/Kev'
  to='kev@remote.example.com/Swift'
  type='groupchat'>
  <body>[[Unclassified]] It's getting warm in here.</body>
</message>

```

Because this is fire and forget mode, the MUC now MUST NOT send the message back to the proxy, but MUST send to the other occupants.

Listing 21: MUC sends the message to the other occupants

```

<message
  from='jabberchat@talk.example.com/Kev'
  to='curtis@example.com/Swift'
  type='groupchat'>
  <body>[[Unclassified]] It's getting warm in here.</body>
</message>

```

#### 5.4 Administration Use Cases

To perform administration of the MUC, connect directly to the MUC and follow the standard process.

## 6 Business Rules

- To avoid complexity of protocol, the MUC MUST NOT modify the nick of a user joining from a proxy - if their JID is unacceptable, the join must instead fail (this simplifies the passing of status codes between the MUC and the proxy).
- Similarly to avoid complexity and round-trips, nick-changing is not allowed for users connected through a proxy. If a user attempts to change their nick, the proxy MUST return a <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas' /> error.

## 7 Security Considerations

This allows a MUC mirror to proxy for another JID, so should only be deployed in scenarios where either the proxy service is trusted, or it is known that the users of the proxy service are in the same security domain as the proxy service.

## 8 IANA Considerations

None.

## 9 XMPP Registrar Considerations

Needs a namespace.

## 10 XML Schema

When advanced.