TigaseDoc

Release 0.1

Tigase, Inc.

Jun 21, 2023

CONTENTS

Welcome to Tigase Multi User Chat component guide

CHAPTER

ONE

OVERVIEW

Tigase MUC Component is implementation of XEP-0045: Multi-User Chat which provides support for multi user chats to Tigase XMPP Server. This component also supports XEP-0313: Message Archive Management protocol for easier retrieval of MUC room chat history.

CHAPTER

TWO

TIGASE MUC RELEASE NOTES

2.1 Tigase MUC 3.3.0 Release Notes

- Rework permission checker (ACL) to add graceful fallback to hidden room if possible; add abstrac muc test class and tests based on it; #muc-151
- Fix memory leak in self-ping-monitor (#muc-150) and improve collections synchronisation in module
- Fix discovery module (Room items could be returned if available but it's advised to not return it by default and only return plain result without error)

2.2 Tigase MUC 3.2.0 Release Notes

2.2.1 Major Changes

- Bring MUC specification support up to date
- Improve handling of multiple user session using same nickname
- Fixes and improvements to ad-hoc scripts

2.2.2 All Changes

- #muc-133: Add component option to let only admins create rooms
- #muc-134: Better MUC Converter log
- #muc-136: MUC specification supported by Tigase MUC is out of data
- #muc-137: Add support for <iq/> forwarding with multiple resources joined
- #muc-138: tigase@muc.tigase.org kicks my clients if I use them both
- #muc-139: Create script to (mass) delete MUC rooms
- #muc-140: There is no empty <subject/> element for persistent room sent after re-joining
- #muc-141: StringIndexOutOfBoundsException in IqStanzaForwarderModule
- #muc-142: NullPointerException when processing message with subject
- #muc-143: Fix MUC scripts: "No such property: mucRepository for class: tigase.admin.Script151"
- #muc-144: No signature of method: tigase.muc.cluster.RoomClustered.addAffiliationByJid()

CHAPTER

THREE

ANNOUNCEMENT

3.1 Major changes

Tigase MUC component has undergone a few major changes to our code and structure. To continue to use Tigase MUC component, a few changes may be needed to be made to your systems. Please see them below:

3.1.1 Database schema changes

We decided to improve performance of MUC repository storage and to do so we needed to change database schema of MUC component. Additionally we decided to no longer use *in-code* database upgrade to update database schema of MUC component and rather provide separate schema files for every supported database.

To continue usage of new versions of MUC component it is required to manually load new component database schema, see *Preparation of database* section for informations about that.

Moreover we no longer store rooms list and configurations inside `UserRepository` of default Tigase XMPP Server database. Instead we use separate tables which are part of new schema. Due to that it is required to execute converter which will move room configurations from `UserRepository` to new tables. It needs to be executed **AFTER** new database schema is loaded to database.

Note: If you used separate database to store messages history we strongly suggest to use same database for new schema and storage of rooms configurations as well. In other case message history will not be moved to new schema.

In `database` directory of installation package there is a `muc-db-migrate` utility which takes 2 parameters:

-in 'jdbc_uri_to_user_repository'
To set JDBC URI of UserRepository

-out 'jdbc_uri_to_muc_database'
To set JDBC URI of database with loaded database schema.

Tip: Both JDBC uri's may be the same.

Warning: During this opeartion it removes room configurations from old storage.

Examples

UNIX / Linux / OSX

```
database/muc-db-migrate.sh -in 'jdbc:mysql://localhost/database1' -out 'jdbc:mysql://
↪localhost/database2'
```

Windows

```
database/muc-db-migrate.cmd -in 'jdbc:mysql://localhost/database1' -out 'jdbc:mysql://
↪localhost/database2'
```

3.1.2 Support for MAM

In this version we added support for XEP-0313: Message Archive Management protocol which allows any MAM compatible XMPP client with MUC support to retrieve room chat history using MAM and more advanced queries than retrieval of last X messages or messages since particular date supported by MUC

3.1.3 Disabled support for XEP-0091: Legacy Delayed Delivery

In this version we disabled by default support for XEP-0091: Legacy Delayed Delivery. This decision was made due to the fact that usage of XEP-0091 is not recommended any more and should be used only for backward compatibility. Moreover, it added overhead to each transmitted message sent from MUC room history, while the same information was already available in XEP-0203: Delayed Delivery format. For more information see *Enabling support for XEP-0091: Legacy Delayed Delivery*

CHAPTER

FOUR

DATABASE

4.1 Preparation of database

Before you will be able to use Tigase MUC Component you need to initialize this database. We provide few schemas for this component for MySQL, PostgreSQL, SQLServer and DerbyDB.

They are placed in `database/` directory of installation package and named in `dbtype-mucversion.sql`, where `dbname` in name of database type which this schema supports and `version` is version of a MUC component for which this schema is designed.

You need to manually select schema for correct database and component and load this schema to database. For more information about loading database schema look into *Database Preparation* section of *Tigase XMPP Server Administration Guide*

4.2 Upgrade of database schema

Database schema for our components may change between versions and if so it needs to be updated before new version may be started. To upgrade schema please follow instuctions from *Preparation of database* section.

Note: If you use SNAPSHOT builds then schema may change for same version as this are versions we are still working on.

4.3 Schema description

Tigase MUC component uses few tables and stored procedures. To make it easier to find them on database level they are prefixed with `tig_muc_`.

4.3.1 Table tig_muc_rooms

This table stores list of rooms and configuration of rooms.

Field	Description	Comments
room_id	Database ID of a room	
jid	Room JID	
jid_sha1	SHA1 value of lowercased room JID	Used for proper bare JID comparison during lookup. (Not exists in PostgreSQL schema)
name	Room name	
config	Serialized room configuration	
creator	Bare JID of room creator	
creation_date	Room creation date	
subject	Room subject	
subject_creator_nick	Nick of participant who set subject	
subject_date	Timestamp of subject	

4.3.2 Table tig_muc_room_affiliations

Table stores rooms affiliations.

Field	Description	Comments
room_id	ID of a room	References room_id from tig_muc_rooms
jid	JID of affiliate	
jid_sha1	SHA1 value of lowercased affiliate JID	Used for proper bare JID comparison during lookup. (Not exists in PostgreSQL schema)
affiliation	Affiliation between room and affiliate	

4.3.3 Table tig_muc_room_history

Table stores room messages history.

Field	Description	Comments
room_jid	Room JID	
room_jid_sha1	SHA1 value of lowercased room JID	Used for proper bare JID comparison during lookup. (Not exists in PostgreSQL schema)
event_type		For future use, if we decide to store other events as well.
ts	Timestamp of a message	
sender_jid	JID of a sender	
sender_nickname	Nickname of a message sender	
body	Body of a message	
public_event	Mark public events	
msg	Serialized message	

CHAPTER

FIVE

CONFIGURATION

To enable Tigase MUC Component you need to add following block to `etc/init.properties` file:

```
muc () {
}
```

It will enable component and configure it under name `muc`. By default it will also use database configured as `default` data source to store data - including room configuration, affiliations and chat history.

5.1 Using separate storage

As mentioned above, by default Tigase MUC component uses `default` data source configured for Tigase XMPP Server. It is possible to use separate store by MUC component. To do so you need to configure new `DataSource` in `dataSource` section. Here we will use `muc-store` as name of newly configured data source. Additionally you need to pass name of newly configured data source to `dataSourceName` property of `default` DAO of MUC component.

```
dataSource {
    muc-store () {
        uri = 'jdbc:postgresql://server/muc-database'
    }
}

muc () {
    muc-dao {
        default () {
            dataSourceName = 'muc-store'
        }
    }
}
```

It is also possible to configure separate store for particular domain, ie. `muc.example.com`. Here we will configure data source with name `muc.example.com` and use it to store data for MUC rooms hosted at `muc.example.com`:

```
dataSource {
    'muc.example.com' () {
        uri = 'jdbc:postgresql://server/example-database'
    }
}

muc () {
```

(continues on next page)

(continued from previous page)

```
    muc-dao {
        'muc.example.com' () {
          # we may not set dataSourceName as it matches name of domain
        }
    }
}
```

Note: With this configuration room data for other domains than example.com will be stored in default data source.

5.2 Configuring default room configuration

It is possible to define value for every room option by setting it's value to `defaultRoomConfig` as a property:

```
muc () {
    defaultRoomConfig {
        <option> = <value>
    }
}
```

for example:

```
muc () {
    defaultRoomConfig {
        'tigase#presence_delivery_logic' = 'PREFERE_LAST'
    }
}
```

5.3 Enabling and configuring MUC room logging

MUC component supports logging inforamtions about

- joining room
- leaving room
- broadcasting message by room
- setting room chat subject

to HTML, XML or plain text files.

To enable this functionality you need to modify `etc/init.properties` file to enable `muc-logger` in MUC component, like this:

```
muc () {
    muc-logger () {
    }
}
```

By default files are stored in `logs` subdirectory of Tigase XMPP Server installation directory. You may change it by setting `room-log-directory` property of MUC component to path where you want to store room logs.

```
muc () {
    'muc-logger' () {
    }
    'room-log-directory' = '/var/log/muc/'
}
```

We provide default logger for room events, but if you want, you may set your own custom logger. Here we set `com.example.CustomLogger` as logger for MUC rooms:

```
muc () {
    'muc-logger' (class: com.example.CustomLogger) {
    }
}
```

5.4 Disable message filtering

MUC component by default filters messages and allows only <body/> element to be delivered to participants. To disable this filtering it is required to set `message-filter-enabled` property of MUC component to `false`.

```
muc () {
    'message-filter-enabled' = false
}
```

5.5 Disable presence filtering

To disable filter and allow MUC transfer all subelements in <presence/>, `presence-filter-enabled` property of MUC component needs to be set to `false`

```
muc () {
    'presence-filter-enabled' = false
}
```

5.6 Configuring discovering of disconnected participants

MUC component automatically discovers disconnected participants by checking if user is still connected every 5 minutes.

It is possible to increase checking frequency by setting `search-ghosts-every-minute` property of MUC component to `true`

```
muc () {
    'search-ghosts-every-minute' = trues
}
```

It is also possible to disable this discovery by setting `ghostbuster-enabled` property of MUC component to `false`

```
muc () {
    'ghostbuster-enabled' = false
}
```

5.7 Allow chat states in rooms

To allow transfer of chat-states in MUC messages set `muc-allow-chat-states` property of MUC component to `true`

```
muc () {
    'muc-allow-chat-states' = true
}
```

5.8 Disable locking of new rooms

To turn off default locking newly created rooms set `muc-lock-new-room` property of MUC component to `false' by default new room will be locked until owner submits a new room configuration.

```
muc () {
    'muc-lock-new-room' = false
}
```

5.9 Disable joining with multiple resources under same nickname

To disable joining from multiple resources under single nickname set `muc-multi-item-allowed` property of MUC component to `false`

```
muc () {
    'muc-multi-item-allowed' = false
}
```

5.10 Enabling support for XEP-0091: Legacy Delayed Delivery

To enable support for XEP-0091 you need to set `legacy-delayed-delivery-enabled` property of MUC component to `true`

```
muc () {
    'legacy-delayed-delivery-enabled' = true
}
```

5.11 Limiting who can create room

For public installations it's desirable to limit visibility of the room - only domain administrators should be able to create publicly visible room that can be discovered by anyone. Everyone else should only be able to create private rooms. This was implemented in https://projects.tigase.net/issue/muc-133.

The feature is configurable via two options: *hidden-room-creation-acl* and *public-room-creation-acl* and follow ACL options defined for Tigase Server (https://docs.tigase.net/projects/tigase-tigase-mix/en/latest/Configuration.html#setting-acl)

```
muc () {
    mucConfig () {
        'hidden-room-creation-acl' = DOMAIN
        'public-room-creation-acl' = DOMAIN_ADMIN
    }
}
```

CHAPTER

SIX

ROOM CONFIGURATION OPTIONS

In addition to the default Room configuration options defined in the MUC specification Tigase offers following as well:

Tigase MUC Options

- tigase#presence_delivery_logic - allows configuring logic determining which presence should be used by occupant in the room while using multiple-resource connections under one nickname, following options are available:
  - PREFERE_PRIORITY
  - PREFERE_LAST
- tigase#presence_filtering - (boolean) when enabled broadcasts presence only to selected affiliation groups
- tigase#presence_filtered_affiliations - when enabled tigase#presence_filtering is enabled one can select affiliation which should receive presences, following are possible to select from:
  - owner
  - admin
  - member
  - none
  - outcast
- muc#roomconfig_maxusers - Allows configuring of maximum users of room.

Configuring default room configuration in init.properties
For more informations look into *???*

Configuration per-room
Per room configuration is done using IQ stanzas defined in the specification, for example:

```
<iq type="set" to="roomname@muc.domain" id="config1">
    <query xmlns="http://jabber.org/protocol/muc#owner">
        <x xmlns="jabber:x:data" type="submit">
            <field type="boolean" var="tigase#presence_filtering">
                <value>1</value>
            </field>
            <field type="list-multi" var="tigase#presence_filtered_affiliations">
                <value>owner</value>
            </field>
        </x>
    </query>
</iq>
```

CHAPTER

SEVEN

OFFLINE USERS

If user affiliation is marked as persistent (which can be done using admin ad-hoc commands), MUC delivers presence to occupants in name of offline user. MUC generates presence with `extended away` info:

```
<presence from="..." to="...">
    <show>xa</show>
</presence>
```

This presence is sent to occupants, when user goes offline and when persistent occupant is added to room (but he is offline). If persistent user if online in room, then MUC sens real presence of occupant.

7.1 Entering the room

Important: When user is joining to room, he MUST use his BareJID as room nickname!

Example of entering to room.

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

7.2 Messages

Room members marked as persistent are able to send message to room, when they not in room. Message will be treated as sent from online user, and delivered to all occupants.

All groupchat messages will be also sent to offline members if they are marked as persistent.