Networking Assignment Chat Protocol

## Introduction

My chat protocol will work on the basis that a channel (room) can be created by anyone, as long as it doesn’t have the same name as a pre-existing channel. Once a user has created a room, they then become the admin of that room. Admins have their own set of commands available as well as having all the powers of an operator and a standard user.

When designing my protocol I drew inspiration from the IRC protocol (Kalt, 2000).

Description Syntax: ‘**|**’ represents ‘or’ and ‘...’ represents recurring previous keyword.

## Channels

Channels can be created by any user. Once a user creates a channel they will automatically become an admin of that channel.

#### Join Channel

**Summary:** Attempts to join a channel with name channelname. The server returns ok if it was successful in joining, invalidname if the channelname cannot be found and noentry if the channel is private or the client’s connection has been banned from accessing the channel.

**Client Sends:** join <channelname>

**Server Sends:** ok | invalidname | noentry

#### Leave Channel

**Summary:** Attempts to leave a channel, if there is no channel to leave or there are issues with the client’s connection (timeout) the server will return failed, otherwise the server will return ok.

**Client Sends:** leave <channelname>

**Server Sends:** ok | failed

#### Create Channel

**Summary:** Attempts to create a channel on the server, set the client as an admin of that channel and move the client into the channel. The server will return takenname if the channel with that name already exists, invalidmode if the client supplies a mode and that mode is not either private or public, and failed if there was a connection issue when creating the channel, otherwise the server sends ok.

**Client Sends:** create <channelname> | create <channelname> <mode>

**Server Sends:** ok | takenname | invalidmode | failed

## Users

### User Commands

These commands will be available to all users including operators and admins.

#### Set Nickname

**Summary:** A user without a nickname will have a randomly generated set of words associated with their connection as a name. They are given the option to create a nickname instead of this.The server returns nametaken if <nickname> is already associated with another user or has a password. Returns invalidnick if the nickname is too long or contains invalid characters. Returns invalidpass if the client supplies an incorrect password.

**Client Sends:** nick <nickname> | nick <nickname> <password>

**Server Sends:** ok | nametaken | invalidnick | invalidpass

### Admin Specific Commands

The admin will be able to set and remove operator status on users. They will also be able to ban users from entering the room, or time a user out. One important difference between an admin and an operator is that the admin can ban and timeout operators and users, whereas operators can only ban and timeout users below operator level.

You may notice that the admin versions of timeout and ban are prefixed with an ‘a’ character in order for the server to differentiate an operator command from an admin command.

#### Set Channel Mode

**Summary:** <mode> can be either: public or private. A public channel means that anyone can join the channel. A private channel means that you cannot join the channel unless you are invited by an operator or an admin on that channel.

**Client Sends:** setmode <mode> <channelname>

**Server Sends:** ok | invalidmode

#### Invite User

**Summary:** Invites a user of the server to the channel. Returns nouser if the nickname supplied cannot be found on the server.

**Client Sends:** invite <nickname> <channelname>

**Server Sends:** ok | nouser

#### Set Operator

**Summary:** Sets an operator role to a client.

**Client Sends:** op <nickname>

**Server Sends:** ok | alreadyop

#### Remove Operator

**Summary:** Removes an operator role from a client.

**Client Sends:** deop <nickname>

**Server Sends:** ok | notop

#### Ban User/Operator

**Summary:** Bans a user or operator.

**Client Sends:** aban <nickname> <channelname>

**Server Sends:** ok | nouser

#### Kick User/Operator

**Summary:** Kicks a user or operator.

**Client Sends:** kick <nickname>

**Server Sends:** ok | nouser

#### Timeout User/Operator

**Summary:** Times out a user for a timer that the server has set.

**Client Sends:** atimeout <nickname>

**Server Sends:** ok | nouser

### Operator Specific Commands

The role of the operator is to moderate a channel. Operators are optional and are not created when a channel is created. There is no limit to number of operators in a room.

#### Invite User

**Summary:** Invites a user of the server to the channel. Returns nouser if the nickname supplied cannot be found on the server.

**Client Sends:** invite <nickname> <channelname>

**Server Sends:** ok | nouser

#### Ban User

**Summary:** Bans a non-operator user. Returns noperm if the nickname supplied belongs to an admin of the channel.

**Client Sends:** ban <nickname> <channelname>

**Server Sends:** ok | nouser | noperm

#### Kick User

**Summary:** Kicks a non-operator user. Returns noperm if the nickname supplied belongs to an admin of the channel.

**Client Sends:** kick <nickname>

**Server Sends:** ok | nouser | noperm

#### Timeout User

**Summary:** Times out a user for a server set amount of time. Returns noperm if the nickname supplied belongs to an admin of the channel.

**Client Sends:** timeout <nickname>

**Server Sends:** ok | nouser | noperm

## Messages

#### Public Messages

**Summary:** Sends a message to all users in the channel.

**Client Sends:** m <message>

**Server Sends:** ok | nochannel | invalid

#### Private Messages

**Summary:** Sends a message to a recipient(s) on the server. The message can only be viewed by the sender and the recipient(s). Multiple people can receive the message from the sender. Returns nouser if any of the usernames cannot be found on the server and invalid if the message is too long.

**Client Sends:** pm <username> <message> | pm <username> <username> … <message>

**Server Sends:** ok | nouser | invalid

## Bibliography

Kalt, C. (2000). *RFC 2812 - Internet Relay Chat: Client Protocol*. [online] ietf.org. Available at: https://tools.ietf.org/html/rfc2812 [Accessed 22 Nov. 2016].

# 