CS 494 / 594
Internet Draft
Intended status: IRC Class Project Specification
Expires: June 1, 2020

Harriet Adkins, Yasodha Suriyakumar
Portland State University
Jan 1, 2020

Internet Relay Chat Class Project
draft-irc-pdx-cs594-00.txt

Status of this Memo

This Internet-Draft will expire on June 1, 2020.

Copyright Notice

Abstract

This memo describes the communication protocol for an IRC-style client/server system for the Internetworking Protocols class at Portland State University.

1. Introduction

This specification describes a simple Internet Relay Chat (IRC) protocol by which clients can connect to a server, and communicate with each other. The server acts as a centralized control. that "relays" messages that are sent to it, and to other connected users.
Users can join rooms that comprise of groups of users. Any message sent to that room is forwarded to all users currently joined to that room. Any client can create a room, join a room, leave a room, and list rooms available at any time.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].
In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in [RFC 2119].

3. Basic Information

All communication described in this protocol takes place over TCP/IP, with the server listening for connections on port 5056. Clients connect to this port and maintain this persistent connection to the server. The client can send messages to the server over the connection, and the server can reply via the same. The messaging protocol is asynchronous - the client is free to send messages to the server at any time, and the server may send messages back to the client at any time. Especially, both the server and client can terminate the connection at any time for any reason.

4. Messages

4.1. Client message Format

When the server is not accepting connections, a simple “Server is sleeping” message is shown to the client.
When the server is active, the client request messages to the Server follow simple string formats. Space is used as a delimiter to identify the command from the rest of the requests to the server, and hence cannot be used in the commands in a meaningful way, The commands and their respective actions are as specified below.

Date	- Get date from the server
Time	- Get time from the server
Options	- Get all the message options
Exit	- Disconnect from the server
Create	- Create a chat room
DisplayRooms	- Display all the available rooms

DisplayRoomMembers	- Display members of a room
Join	- Join room
Leave	- Leave room
SendMessage	- Send message to all clients in a room

4.2. Server message Format

The server addresses each client by the IP and the port from which the client connected. The server messages also follow simple string formats. All messages to the clients, except for disconnecting are preceded by “>>” symbols. When valid messages are sent from the client, the appropriate action taken by the server is notified to the client. All the error messages to the client are shown with a preceding “Invalid input”.
Server provides the option to disconnect by allowing the user to enter “Exit” in the server’s communication window. Whenever a new client connects to the server, the server responds by creating a new thread to handle communications with that client. The user is notified of the IP address and the port from which the client connected.

5. Label Semantics

Identifying the chat rooms, and messages involves sending and receiving labels. All label rules are the same, and MUST be validated as follows.

- “:” are not allowed in room names (as “:” is reserved as a delimiter for sending messages)
- “,” also serves as a delimiter - no action is taken on empty strings between “,’ symbols
- Space is allowed in room name labels

6. Client messages and usage

6.1. Date

Usage:

Server’s date is requested by entering “Date”. Client is expected to send “Date” as the message packet.

Response:

Server date is sent to the client in “yyyy/MM/dd '' format.

6.2. Time

Usage:

Server’s time is requested by entering “Time”. Client is expected to send “Time” as the message packet.

Response:

Server time is sent to the client in “hh:mm:ss” format.

6.3. Options

Usage:

Options that are allowed for the client. Client is expected to send “Options” as the message packet.

Response:

The available options as shown below is sent to the client in the packet.
>> Your options are: (Date | Time | Create [roomName(,s)] | DisplayRooms |
>> DisplayRoomMembers [roomName(,s)] | Join [roomName(,s)] | Leave [roomName(,s)] | SendMessage [roomName]:[message],[roomName]:[message])
>> Type Options for choices or Exit to terminate connection.

6.4. Exit

Usage:

Clients are allowed to disconnect from the server and all the rooms that they participate in by sending “Exit”.

Response:

Server responds by disconnecting the client connection, and the message “Connection closing.. You are disconnected”. Server does not inform the other clients in the participating room(s) that this client has exited the session.

6.5. Create

Usage:

Clients are allowed to create rooms by sending messages with the format “Create [roomName(,s)]”. The room names can be a comma separated list for multiple rooms. And, the names follow the label rules defined in Section 5.

Response:

Server responds by creating a room with the provided name(s), if no room with the name exists. If the room exists, then it is notified to the client.

6.6. Join

Usage:

Clients are allowed to join rooms by sending messages with the format “Join [roomName(,s)]”. The room names can be a comma separated list for multiple rooms. And, the names follow the label rules defined in Section 5.

Response:

Server responds by adding the member to the room with the provided name(s), if a room with the name exists. If no room exists, then it is notified to the client.
When a client joins a room, the server MUST notify all the other clients in the room that a new client has joined the room.

6.7. DisplayRooms

Usage:

Clients are allowed to list all rooms by sending the message “DisplayRooms”.

Response:

Server responds by sending all the available rooms as a comma separated list at the requested time.

6.8. DisplayRoomMembers

Usage:

Clients are allowed to list all rooms by sending the message “DisplayRoomMembers ”. The room names can be a comma separated list for multiple rooms. And, the names follow the label rules defined in Section 5.

Response:

Server responds by sending the room member’s names as a comma separated list for all the valid rooms at the requested time.

6.9. Leave

Usage:

Clients are allowed to leave rooms by sending messages with the format “Leave [roomName(,s)]”. The room names can be a comma separated list for multiple rooms. And, the names follow the label rules defined in Section 5.

Response:

Server responds by removing the member from the room with the provided name(s), if a room with the name exists. If no room exists, then it is notified to the client.
When the client leaves a room, the server MUST notify all the other clients in the room that the client has exited the room.

6.10. SendMessage

Usage:

Clients are allowed to send messages to rooms with the format “SendMessage [roomName]:[message],[roomName]:[message]”. The room name and message combination can be a comma separated list for multiple rooms. And, the names follow the label rules defined in Section 5.

Response:

Server responds by sending the messages to all the clients in the appropriate room(s) in the format “[Client name] in room [roomName] says [message]“. This response is done, if a room with the name exists. If no room exists, then it is notified to the client initiating the sendMessage request.

7. Server messages and usage

Exit

Usage:

Server is allowed to disconnect by entering “Exit”.

Response:

The server MUST notify all the active clients that the server is quitting the session, and the client sockets as well the server’s open socket are closed.

8. Error Handling

When the server and client are sending and receiving messages, they MUST detect when the socket connection linking them is terminated. When the server detects that the client connection is lost, the server MUST remove the client from all the participating rooms. If the client detects that the server connection is lost, it MUST consider itself disconnected from the services, and MAY choose to reconnect at a later time.
As discussed in Section 6., the server notifies the client in case of invalid messages from the clients.

9. "Extra" Features Supported

No other “extra” features are supported.

10. Conclusion & Future Work

This specification provides a generic message passing framework in which multiple clients can communicate with each other through a central forwarding server. Features such as private messaging, and secure messaging can be added using this framework.

11. Security Considerations

Messages sent using this system are not secure. They can be intercepted, and inspected. Also, the server intercepts all the messages sent between the clients that use this service.

12. IANA Considerations

None

Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

13. Acknowledgments