James Ross
RFC Documentation
Expires: June 2018

Abstract

This draft describes the communication messages and protocol for an IRC client-server application and defines the types of communication between the two.

Table Of Contents

1. Basic Information

The communication between the client and the server process takes place over port TCP/IP 50000 and holds a persistent connection unless the client or server exits by closing or breaking the socket connection. Messages can be sent or received on both the client and server at any given time.

The server will not allow two users with the same name to connect. It will leave the port open for the client to retry a name until it gets one. Since the server does not currently store user credentials no password is associated with logging in. The client must handle keeping track of friend and block lists if desired until the server has persistent logon credentials and stores that client information on or offline.

If the client or the server receives a message of 0, the connection will be considered closed and the server and client must close the socket.

2. Message Format Information

2.1 Generic Message Format

The message format has 3 generic fields.

Field 1: Client name (string, **Client only**)
Field 2: Opcode (8-bit integer)
Field 3: Payload (uint8_t*)

All messages to or from client/server end with a carriage return ('\r')

2.1 Opcodes

RC_JOIN 0x7 , request from and respond to clients joining a room.
RC_EXIT 0x8 , Inform each other during exit of programs.
RC_LOGON 0x11 , Used during the initial logon process between client and server.
RC_LEAVE 0x15 , request from and respond to clients leaving a room.

RC_MSG 0x16 , client requests a message to a room, server distributes.
RC_RUL 0x17 , client requests a list of all users in a room. Server responds.

RESERVED_Z 0x0 , the opcode 0x0 is unavailable to the client-server communication.
RESERVED_CR 0x13 , Since '\r' ends communication messages, not using as opcode.

3. Client Messages to Server

3.1 RC_LOGON, initial log in message to server.

3.1.1 Usage

After initial connection with the server, the server will obtain the new clients socket
information and store it. A following message must be sent from the client to the server with the opcode RC_LOGON which will contain the desired client name string from field 1 as described in section 2.1.

The server must ensure no other client currently exists with that name before finishing adding the client. If the name already exists, the server's reply will identify it using this same opcode.

3.1.2 Field Descriptions

Field 1: client name (string)
Field 2: opcode (8-bit integer)

3.2 RC_MSG, Send a message to currently displayed room.

3.2.1 Usage

The client must send the room name the message is for and provide the message being sent.

The client should update its room history and hold off display until a message from the server with opcode RC_MSG is received containing the message to the room.

3.2.2 Field Descriptions

Field 1: client name (string)
Field 2: opcode (8-bit integer)
Field 3: room name (string)
Field 4: Message (string, max 255 bytes)

3.3 RC_JOIN, Request to and join a given room.

3.3.1 Usage

The client will send a request to join a given room. If the room requested does not exist, the room will be created and the client will be placed as the first person in that room, otherwise the client will join the room provided no errors.

The server upon receiving the message must check if the room exists, if not the server must create it and include the client. If the room already exists, the server will then check the number of users in the room and ensure it is not full. If it is full, the success/failure field in the servers reply will be true/false respectively. Otherwise, the server will reply to the client, letting them know they successfully joined the room and can update their local information. The client should not update their users data until the reply from the server.

3.3.2 Field Descriptions

Field 1: client name (string)
Field 2: opcode (8-bit integer)
Field 3: Name of room to join (string)

3.4 RC_LEAVE, Request to and leave a given room.

3.4.1 Usage

The client will send a request to leave a given room. The client should ensure the user is actually currently in the room. The server checks if the client is actually in the room and responds with failure in the result field of the server

reply . The client will do nothing if there is nothing to be removed.

If the client is in the room, they will be removed from the room's user list and a response will return to the client letting them know they are no longer in the room and can update their local data structures.

If all the rooms are left for a client, the handling of the user must be done by the client until the user wants to join a server on the room. A default room name for this behavior is recognized by the server and client and is named "void" but is not allocated on the server as a room.

3.4.2 Field Descriptions

Field 1: Client name (string)
Field 2: opcode (8-bit integer)
Field 3: room name to leave (string)

3.5 RC_EXIT, Client is currently shutting down connection gracefully.

3.5.1 Usage

The client sends the opcode RC_EXIT when the client wants to gracefully disconnect from the server.

The server must remove all data relating to the client (until persistent client username/password is implemented) including its spot in room lists. A response to the client will be sent prior to closing the socket on the server end letting the client know they have successfully exited from the server and can close the connection on the client side.

3.5.2 Field Descriptions

Field 1: Client name (string)
Field 2: opcode (8-bit integer)

3.6 RC_RUL, Privately message another client.

3.6.1 Usage

The client sends this message requesting all users inside a given room name. The server will reply with a series of messages until each user name is sent to the server. The client should then display or store the names for the user.

3.6.2 Field Descriptions

Field 1: client name
Field 2: opcode
Field 3: room name

4. Server Messages to Client

4.1 RC_LOGON, reply to initial logon request.

4.1.1 Usage

Upon receiving the RC_LOGON message, the server must read the client name and guarantee the client name is not taken. If the client name is taken, the result field will be returned with a 0 and the client can end connection or try again.

Upon success, the server saves and stores all relevant client information for further communication.

Maximum name size is 10 characters. 11 including a '\0' value for the string.

4.1.2 Field Descriptions

Field 1: RC_LOGON (uint8_t)
Field 2: Result (uint8_t)

4.2 RC_MSG, handle a new message from client in current room.

4.2.1 Usage

When the server receives the RC_MSG command from the client it will ensure the room name is valid and that the room exists. The server also must make sure the client is in the room before sending the message. This should also be checked by the client to prevent server overhead.

If the room is valid the message will be broadcasted to every client in the room containing the sender's name, what room it is for and the message that was sent for the client to handle or display as they choose.

4.2.2 Field Descriptions

```
Field 1: RC_MSG    (uint8_t)
Field 2: Sender's name (string)
Field 3: Room name      (string)
Field 4: Message        (string)
```

4.3 RC_JOIN, Adds client to a room, updates clients in that room.

4.3.1 Usage

Upon receiving RC_JOIN messages from the client, the server will make sure the room name provided is valid based on the room name length max of 20 characters, 21 including the ‘\0’ value at the end of the string.

If the room is a valid name, the server will create the room if it does not yet exist or place the client in the room that does exist.

The server will reply with a success or failure result, the room name that was joined and the number of users in the room.

Failure will also result when the room has reached its max user limit defined by _R_USR_MAX which is set to 100 users per room.

4.3.2 Field Descriptions

Field 1: RC_JOIN (uint8_t)
Field 2: number of users (uint8_t)
Field 3: Room name (string)
Field 4: result (bool)

4.4 RC_EXIT, Reply to a request for graceful disconnect.

4.4.1 Usage

The server will ACK the client of the exit request before shutting down the socket. A successful exit includes full deallocation of the client from the server due to persistent client information not being implemented. The client should exit upon receiving a response to make sure the server received the message, but is not required.

4.4.2 Field Descriptions

Field 1: RC_EXIT (uint8_t)

4.5 RC_LEAVE, confirming client leaving a room on server end.

4.5.1 Usage

The client sends a RC_LEAVE message type to remove themselves from the room. The room void is reserved and can be used by the client locally to handle moments where the server does not have the client placed in the room and can apply functionality to the default room which will always hold 0 users on the server side.

The result field lets the client know if they were successfully removed from the servers room information and the client should not be considered removed from the room unless the response is a success.

4.5.2 Field Descriptions

Field 1: RC_LEAVE (uint8_t)

Field 2: room name (string)
Field 3: result (bool)

4.6 RC_RUL, Sending list of all active rooms to a client by request

4.6.1 Usage

When receiving the first RC_RUL message, the server will initiate sending all the users from the room name provided. If there is no one in the room it will send the final RC_RUL reply message indicating that no one is in the room.

If there are people in the room, the server will response with all description fields sending a user name one at a time until all users have been sent. If all users has been sent or the room was empty the user name field is omitted and holds the terminating character '\r'.

If multiple requests are made simultaneously by the same client, they will be handled in order of request.

4.6.2 Field Descriptions

Field 1: RC_RUL
Field 2: User name

5.0 Error Handling

If for some reason the client or server crashes and cannot execute the RC_EXIT protocol, the socket on the remaining side of the connection will receive a empty message. Because of this, any message that gets sent to the client or server must behave as a disconnect and execute exit operations for that specific socket.

6.0 Security

There is no guaranteed encryption or security for this protocol. This server and protocol will be vulnerable to some attacks. No encryption process is used.