Internet Technology

(Protocol Specification)

Protocol Specification:

Protocol	Description	Example
INIT	Server confirms connection with client.	S:INIT Welcome to the server!
IDNET	Clients identify themselves by assigning a username.	C:IDENT user1 S:OK IDENT user1 S:JOINED user1
BCST	Broadcasting message to the connected clients.	C:BCST hello S:OK BCST hello S:BCST user1 hello
OK	Server acknowledgement for client requests.	C:IDENT user1 S:OK IDENT user1 S:JOINED user1
JOINED	Informs other clients of a user that has connected to the server.	
QUIT	Sent by a client to leave the server.	C:QUIT S:OK GOODBYE Broadcast: S:LEFT <username>
GOODBYE	Sent by the server to send a goodbye message to the client.	
USERS	Allows clients to get a list of the connected clients from the server.	C:USERS S:OK USERS user1:user2
PING	Periodic ping sent to the client to check connectivity.	Happy Flow: S:PING C:PONG Unhappy Flow: S:PING S:DSCN Pong timeout S:FAIL05 Ping without pong
PONG	Reply to the PING received from the server to confirm connectivity.	
DSCN	Sent by the server to disconnect a client.	
LEFT	Sent to clients to inform them that a user has left the server.	

		Broadcast: S:LEFT <username>
PM	Allows a client to send an encrypted private message to another user.	C:PM user2 <encrypted_msg> S:OK PM user2 <encrypted_msg> To recipient: S:NPM user1 <encrypted_msg>
NPM	Allows the server to forward an encryped private message to the recipient.	
PM_PUBKEY	Allows the client/server to share a public key.	C: PM_PUBKEY <public_key> To recipient: S: PM_PUBKEY <sender_username> <public_key> C: OK PM_PUBKEY
PM_SESSIONKEY	Allows clients to exchange session keys.	C: PM_SESSIONKEY <recipients_username> <encrypted_session_key> C: PRIVATE_MESSAGE <recipients_username> <encrypted_message> S: OK PRIVATE_MESSAGE <recipients_username> <encrypted_message> To recipient: S: PM_SESSIONKEY <senders_username> <encrypted_session_key> C: OK PM_SESSIONKEY S: RECEIVE_PRIVATE_MSG <senders_username> <encrypted_message>
FAIL00	Unknown command.	
FAIL01	User already logged in.	
FAIL02	Username entered has an invalid format or length.	
FAIL03	User must login first.	
FAIL04	User cannot login twice.	

FAIL05	No response to ping.	
FAIL06	Client is the only connected user.	
FAIL07	User does not exist.	

New implementations

1. Connected clients:

This protocol is used to send a list of connected of connected clients. The chat server can provide a list of connected users upon request. If there are no connected users, the server will return an error. The list of usernames will be separated by a dash, so usernames cannot contain dashes.

Happy flow:

C: USERS

S: OK USERS <Username1>:<Username2>:<Username3>

Error message:

S: FAIL06 YOU ARE THE ONLY CONNECTED USER

2. Private messages:

The user can send and receive private messages by sending the following command.

Happy flow:

C: PM < Receiver> <message>

S: OK PM <Receiver> <message>

To the receiver:

S: PRIVATE MESSAGE <Sender> <message>

Error message:

S: FAIL07 USER DOES NOT EXIST

3. Encryption:

Public Key Initial exchange:

Happens automatically when a user joins the chat server, copies of each client's public keys will be stored on the clients, and the clients will update the list when users join or leave the server.

Happy Flow:

C: PM_PUBKEY <public_key>

To recipient:

S: PM_PUBKEY <sender_username> <public_key>

C: OK PM_PUBKEY

Session key and message exchange:

If the session key has already been exchanged, then it is not sent again. Once a user leaves the chat server their session key is automatically erased. Messages are encrypted using the session key only (asymmetric encryption). The session key is encrypted using the recipient's public key (asymmetric encryption) before being sent to them, the recipient then decrypts it using their private key. The session key is randomly generated and sent automatically on the start of a session.

Happy Flow:

C: PM_SESSIONKEY <recipients_username> <encrypted_session_key>

C: PRIVATE_MESSAGE <recipients_username> <encrypted_message>

S: OK PRIVATE_MESSAGE <recipients_username> <encrypted_message>

To recipient:

S: PM_SESSIONKEY <senders_username> <encrypted_session_key>

C: OK PM_SESSIONKEY

S: RECEIVE_PRIVATE_MSG <senders_username> <encrypted_message>

Error Messages:

S: FAIL03 USER MUST LOGIN FIRST

S: FAIL07 USER DOES NOT EXIST

S: FAIL08 CANNOT SEND PRIVATE MESSAGE TO YOURSELF