CS594 Portland State University

Internet Draft May 8th, 2023

Intended status: IRC Class Project Specification

Expires: June 2023

RFC Format Framework

Status of this Memo   
  
This Internet-Draft is submitted in full conformance with the   
provisions of BCP 78 and BCP 79. This document may not be modified,   
and derivative works of it may not be created, except to publish it   
as an RFC and to translate it into languages other than English.   
  
Internet-Drafts are working documents of the Internet Engineering   
Task Force (IETF), its areas, and its working groups. Note that   
other groups may also distribute working documents as Internet-   
Drafts.   
  
Internet-Drafts are draft documents valid for a maximum of six months   
and may be updated, replaced, or obsoleted by other documents at any   
time. It is inappropriate to use Internet-Drafts as reference   
material or to cite them other than as "work in progress."   
  
The list of current Internet-Drafts can be accessed at   
http://www.ietf.org/ietf/1id-abstracts.txt   
  
The list of Internet-Draft Shadow Directories can be accessed at   
http://www.ietf.org/shadow.html   
  
This Internet-Draft will expire on June 1, 2023.   
  
Copyright Notice   
  
Copyright (c) 2022 IETF Trust and the persons identified as the   
document authors. All rights reserved.   
  
This document is subject to BCP 78 and the IETF Trust's Legal   
Provisions Relating to IETF Documents   
(http://trustee.ietf.org/license-info) in effect on the date of   
publication of this document. Please review these documents   
carefully, as they describe your rights and restrictions with respect   
to this document. Code Components extracted from this document must   
include Simplified BSD License text as described in Section 4.e of

Internet-Draft Internet Relay Chat Class Project May 2022   
  
  
the Trust Legal Provisions and are provided without warranty as   
described in the Simplified BSD License.

Abstract   
  
This small IRC project is similar to the [IRC](https://datatracker.ietf.org/doc/html/rfc1459#page-14). This project consists of two applications a central server and many clients. The server will serve as a central point for the clients to send messages to each other.

1. Introduction  
     
   This specification describes an IRC protocol and implementation that allows developers to
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.   
     
   In this document, the characters ">>" preceding an indented line(s)   
   indicates a statement using the key words listed above. This   
   convention aids reviewers in quickly identifying or finding the   
   portions of this RFC covered by these keywords.
3. Server

3.1 What's a server

A server is the central point of the IRC application. Any host can be a server but there should only be one. In fact, for the scope of this project, the server will be hosted on the same machine as the clients. More time will be needed to configure the server to be hosted on the cloud. The server for this project will be written in Haskell.

3.2 What's a client

A client can be any desktop or webpage that connects to the server. It communicates with the server via specified JSON messages and websockets will be further detailed in later sections. The client for this implementation will use Python.

3.3 What's a user

In our app a user is anyone that is able to use a client that establishes a connection with the server. A user can be anyone as long as they give a unique name to the server.

3.3.1

A user does not require password credentials at this time and they will not be persisted after server power cycles.

3.4 What's a room

A room is a collection of users. Each user is able to write to a room that broadcasts to everyone else in the room.

2. The IRC Specification

2.1 Websockets

The server and client will be connected via long lived websockets. This allows the server to broadcast messages to any client connected to the server.

2.2 Interactions

* Request – clients will format a request to the server and each request will be specified in 2.4
* Response – the server may respond to the request with extra data for the client to keep track of.
* Broadcast – some requests will trigger a broadcast to certain users. The scope of the broadcast will depend on the request.

2.3 Payload format

The server will accept payload information in the form of JSON text.

2.4 Available Actions

2.4.1 Connect to Server

The user will connect to the server and send a message for a username. If the username is taken the client will receive a message back and the connection will be terminated. The client can establish another connection and try again.

2.4.1 Login  
  
Client Request

{“request”:”LOGIN”,“username”:<username>}

<username> is the handle the user wishes to assign himself. The name must not contain punctuation, whitespace nor be empty.  
  
Server Response

{“response”:”USER\_LOGGED\_IN”, “userId”:<userId>}  
<userId> is the UUID assigned to the user upon successful login.

2.4.2 Create Room

The client will be able to create a room for others to join.  
  
 Client Request  
 {“request”:”CREATE\_ROOM”,“room\_name”:<roomname>}

<roomname> will be the name the user wishes to assign to the room. There are no checks on room name.  
  
Server Response

{“response”:”ROOM\_CREATED”, “roomId”:<roomId>}  
<roomId> is the UUID assigned to the room for the client to keep track of.

2.4.3 Join Room

If a room exists, a client can join that room.

Client Request  
{“request”:”JOIN\_ROOM”,“roomId”:<roomId>}

<roomId> will be the id of the room the client received either upon creating a room or getting a list of rooms.  
  
Server Response

{“response”:”ROOM\_JOINED”, “roomId”:<roomId>, “users”:<users>}  
  
<roomId> is returned so the user can know which room they were allowed to join.

<users> are the list of users that are also in the room.

Broadcast Response

{“broadcast”:”USER\_JOINED\_ROOM”, “roomId”:<roomId>, “userId”:<userId>, “username”:<username> }  
  
<roomId> is the id of the room the user is joining.

<userId> is the id of the user who is joining the room.

<username> is the username of the user who is joining.

2.4.4 Leave Room

A client will have the option to leave a room to stop receiving broadcasts directed to users of a room.  
  
Client Request  
{“request”:”LEAVE\_ROOM”,“roomId”:<roomId>}

<roomId> will be the id of the room the client received either upon creating a room or getting a list of rooms.  
  
Server Response

{“response”:”ROOM\_LEFT”, “roomId”:<roomId>}  
  
<roomId> is returned so the user can know which room they were allowed to join.

Broadcast Response  
  
{“broadcast”:”USER\_LEFT\_ROOM”,”roomId”:<roomId>,”userId”:<userId>}  
<roomId> is the id of the room that the user is leaving.  
<userId> the userId that left the room.

2.4.5 List All Rooms

A user will have ability to request all the rooms they can join.  
  
Client Request  
{“request”:”LIST\_ALL\_ROOMS”}   
  
Server Response

{“response”:”LIST\_OF\_ROOMS”,“rooms”:<rooms>}

<rooms> will be a list of objects of rooms that include roomId and room name.

2.4.6 List Room Members

A user will have the ability to see everyone in the room after they join.  
  
Client Request  
{“request”:”LIST\_ROOM\_MEMBERS”, “roomId”:<roomId>}

<roomId> will be the id of the room the client received either upon creating a room or getting a list of rooms.

Server Response

{“response”:”LIST\_OF\_USERS”,”roomId”:”<roomId>”,“users”:<users>}

<roomId> is the roomId associated with the list of users. The users are structured JSON objects containing an Id and the username.

2.4.7 Send a message to room

A user can send a message targeted to a room which will be broadcasted to everyone else in the room.  
  
Client Request  
{“request”:”SEND\_ROOM\_MSG”, “roomId”:<roomId>, “data”:<data>}

<roomId> is id of the room the client wants to send a message to.  
<data> is the msg data the room will receive.  
  
Server Broadcast

{“broadcast”:”ROOM\_MESSSAGE”, “roomId”:<roomId>, “userId”:”<userId>”, “username”:”<username>”,“msg”:<msg>}  
  
<roomId> to let client know the room that is targeted.

<userId> is the user who sent the msg

<username> a convenience field to let the client know the username of the sender.

<msg> is the message the user sent.

2.4.8 Logout

To logout a user will send a logout message to the server and the server will close the connection.  
  
Client Request  
{“request”:”LOGOUT”}  
  
Server Response

{“response”:”USER\_LOGGED\_OUT”, “userId”:<userId>}  
  
<userId> isn’t necessary for this response but it’s returned anyway. Just for extra confirmation.

Broadcast Response

{“broadcast”:”USER\_LOGGED\_OUT”,”userId”:<userId>,”rooms”:[<roomId>]}

<userId> the id of the user who has logged out.

<roomId> the room the user was in. The client can remove the user based on this notification.

2.5.9 Destroy Room

A user can destroy a chat room and all users in that room will be kicked out of the room.  
  
Client Request  
{“request”:”DESTROY\_ROOM”, “roomId”:<roomId>}  
  
Server Response

{“response”:”ROOM\_DESTROYED”, “roomId”:<roomId>}  
<roomId> the id of the room that was destroyed for confirmation.  
  
Broadcast Response

{“broadcast”:”ROOM\_DESTROYED”,”roomId”:<roomId>}  
  
<roomId> the id of the room that was destroyed.

4. Glaring omissions

4.1 No encryption

Communication between server and client will not be secure.

4.2 Information will not persist since not using a database

Because no information will be persisted to the database, should the server power cycle all information will be lost. No logs of text, no user information or anything.

4.3 No authentication/credentials

For this project there will be no authentication nor authorization systems for users.