Camelot Protocol Document

A description of how the system protocol will behave. Created and maintained by Ian Howell, Hunter Mathews, Illya Starikov, William Thurman, Zachary Wileman. *Server Team #1*

Last Revised: *March 14, 2017*
Illya Starikov

Michael Gosnell
Immediate Supervisor
mrghx4@mst.edu

Status: **Validated**
Version: **1.1**

Contents

In essence, our server will act as an IRC-esque chatroom. Because we are starting with the Minimal Viable Product (MVP), it will not particularly have all the features of an IRC server; also, because we have to provide a JSON-based API, that alters the design drastically. This document will provide a sufficient outline on the authentication process, the basic objects (user, channel, message), and other miscellanies. Required fields are marked by a red asterisk (*), and possibly required items are marked with a blue asterisk (*).

The server will be asynchronous. The messages can be sent from the client at anytime, and will be handled accordingly. The messages should also be able to be received at anytime. It is up to the client to add listeners to be able to receive said messages.

1 Authentication

At the time of the authentication (i.e. the login process) a few item will have to be provided.

1.1 Input

Server Location* The server location must be provided to get access to the server. Duh. **String.**

Server Password* An optional password might be required. **String.**

User* A unique identifier to represent the user. If submitted, and not unique to channel, a randomized one will be returned. Can only consist of a mixture of numbers $[1 \cdots 9]$, letters $([a \cdots z]|[A \cdots Z])$ or underscores (_). **String. String.**

Channels *After* authentication (i.e. after successfully joining the server), channels can be joined. Refer to Section 2 for details. **Array of Objects `[{ Channel, Password }, ⋯ ]`**

1.2 Output

Session Key Will be used in future development. **Array.**

Username Will be used to verify if submitted username was allowed. **String.**

Channel List Success Will return an a array of objects to specify the success of joining the channels. **Array of Objects `[{Name, Success Status, [Users, ⋯] }, ⋯].`**

2 Channel

The channels is the actual location of the chat room. All messages posted by the user(s) will appear here. A channel can act as a direct message mechanism as well, by simply having two users.

Name The name of the channel. The name of the channel must be unique. Can only consist of a mixture of numbers $[1 \cdots 9]$, letters $([a \cdots z] | [A \cdots Z])$ or underscores (_) **String.**

Password If there a password, specify it. If there is no password, any submissions (including SQL injections) will be accepted. **String.**

3 Message

A message is a simple object with what you'd expect a message to have. Simple message object will be sent and received (with all the contents below).

Timestamp The time *the client* sent this message. A JavaScript format time object should be sent. **String.**

Sender* The *username* of the sender. **String.**

Message Actual message contents. No format characters will be accepted (i.e. `\t`, `\n`, or `\r`). **String.**

Channel What channel the message was posted on. **String.**

4 Edit(s)

This section is dedicated compatibilty-breaking edit(s).

Illya Starikov – Anonymity Discontinued

Not only is true anonymity hard to implement and frustrating to maintain, a lot of the time it is only a hassle for users and programmers. The only people gaining from anonimity are typically trolls or abusers of the system. From this point forward, the bare minimum one can provide is a username. It is to the clients if they wish to make a randomized ID, but it should be displayed. *Changed on February 28, 2017.*