The WS_2017 protocol

Max Hackinger

March 24, 2017

Contents

Abstract

This document will Introduce and define the WS_2017 protocol. The WS_2017 protocol uses the POP3 protocol as a reference.

1 Session States

The session states in the current draft are directly lifted from the POP3 protocol.

The session goes through several states during it's life time. After the TCP connection is established and the server sends a greeting and the session enters the AUTHORIZATION state. By transmitting a user name to the server with the **uname** command the client authorizes it's self and the session can advance to the TRANSACTION state. In the TRANSACTION state the client can execute commands till the client sends the **cquit** command, that then moves the session in to the UPDATE state. In this state the server releases all resources from the TRANSACTION state and says goodbye and the TCP connection is closed.

2 Basic Overview

The WS_2017 service is started by the server listening to port 1030. To start a session the client establishes a TCP connection with the server and the server sends a greeting to the client. Commands are exchanged between the client and server till the connection is closed or aborted.

3 Commands

All commands are case insensitive and made up entirely of ASCII characters. Commands always have exactly one **keyword** followed by none or more **arguments**. The keyword is never longer then 5 characters. Commands are always terminated with a CRCF (Carriage Return: \r, New Line: \n). Commands will either be answered with a positive response confirming that the command has been understood and processed or negative response pointing out what is wrong.
A positive response has a '+OK' followed by the command that was successful. A negative response has a '-ERR' followed by the command that failed and an argument that either is a message with what went wrong or a suggestion for change that is relevant to the keyword.
If the entered command is badly formatted the Server should return:

s: -ERR '<command> is not a properly formatted command'

If the entered command does not match a valid command, the server should return:

s: -ERR 'entered command does not exist'

4 AUTHORIZATION State Commands

Once the session has gone in to the AUTHORIZATION state, the server will be expecting user name to identify the client by. This is done by sending a message with the desired user name to the server. If the name is already present in the server (i.e there is already a client connected with that name), the server returns a negative response. If the name given to the server is unique the server confirms it with a positive response. When a user disconnects from the server the user name is removed.

registering a name:

> c: uname '<name>'
> s: +OK 'you are <name>'

own username entered:

> c: uname '<name>'
> s: -ERR same username entered'

username already taken:

> c: uname '<name>'
> s: -ERR uname '<name_suggestion>'

5 TRANSACTION State Commands

5.1 Ping/Pong

in regular intervals server and client should exchange pings to make sure that they are still connected, if there is no response in a defined time the connection should be disconnected.

server sends ping:

s: ping
c: +OK pong

5.2 Changing user name

To change user name the same command is used as to enter the original user name in the AUTHORIZATION State

c: uname <new_name> s: +OK you are <new_name>

5.3 Get all user names

To be able to send a message or find to an other user one needs to know the name of the other users.

c: cgetu
s: +OK <user0> <user2> <user2> ...

5.4 Chat

When chatting, the server acts as a relay between two clients. When the message arrives at the destination client, the recipient automatically sends back a message (with the chatr command) to the server that then relays a message to the sending client that the message was received.

sender:

c: chatm '<sender_name>' '<recipient_name>' '<message>'
s: +OK 'message relayed'

recipient:

s: chatm <sender_name> <recipient_name> '<message>'
c: chatr <sender_name> <recipient_name>

c: cgetu

6 UPDATE State Commands

6.1 Quit

When the client wants to terminate the connection to the server, the client uses the quit command which leads the session from the TRANSACTION state to the UPDATE state. In this state the server ends tasks related to the client in a safe manner.

c: cquit
s: +OK 'terminating tasks and disconnecting'