The WS_2017 protocol

Max Hackinger

March 6, 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 The WS_2017 protocol

1.1 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, the client needs to authorize it's self in the AUTHORIZATION state. Once this is done the server collects info pertaining to the client and the session enters the TRANSACTION state. The client can execute commands till the client sends the QUIT 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.

1.2 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.

1.3 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.

1.4 AUTHORIZATION state commands

Once the session has gone in to the AUTHORIZATION state, the server will be expecting a name to identify the client by. If the server recognizes it as a name that has already been used before and the IP address is the same as the client that last used it, it will return a positive message and all relevant information for the client. If it is a new name the server will create a new entry in the list of names. If the name is a previously used name and the IP address does not match, the server will return negative message.

registering a name:

c: rname <name>
s: +OK you are <name>

recognizing client:

c: rname <name>
s: +OK welcome back <name>

negative response:

c: rname <name>
s: -ERR <name> is already taken by another client

Alternatively the server could remove used names after the client disconnects. When a client connects to the server the client has to send a message for the server to identify it by. 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 message. If the name given to the server is unique the server confirms it with a positive response.

registering a name:

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

negative response:

c: rname <name>
s: -ERR '<name> is already taken by another client'

1.5 Transaction state commands

1.5.1 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.

client sends ping:

c: ping
s: +OK pong

server sends ping:

s: ping
c: +OK pong

1.5.2 change name

c: cname <new_name>
s: +OK 'name changed from <old_name> to <new_name>'

1.5.3 chat room

1.5.4 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>

1.6 UPDATE commands

1.6.1 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: quit
s: +OK 'terminating tasks and disconnecting'