Socket Server Adapter Communication Protocol

TEMA Unix package

Antti Kervinen

December 21, 2010

1 Introduction

This document specifies the communication protocol implemented in TEMA test engine's socket server adapter.

When the TEMA test engine wants to execute a keyword, it calls

`sendInput(`*`keyword string`*`)`

method of the adapter chosen for the test run. The main task of the socket server adapter is to forward these calls to remote clients over TCP/IP.

Inititally, the socket server adapter listens to a port for TCP/IP connections. A client (the real adapter) connects to the port, says "HELO" and waits for acknowledgement. When received, the client goes to a loop where it first requests a keyword, then executes it, and finally reports the result to the adapter.

The client and the server talk by turns. That is, after sending a message one must receive a message. Initially, client sends and the server receives.

The protocol is designed to be used easily over a telnet connection. Messages are in plain text, one message per line.

2 Messages

Message are plain text and are separated with linefeeds. Carriage returns are also allowed: all whitespace should be stripped from the end of the messages. Some messages may have parameters, which are separated from the messages with one space character. The following messages can be sent.

	From client to server
`HELO`	The first message from a client to a server after establishing a connection.
`GET`	Keyword request
`PUT`	Parameter required, either `TRUE` or `FALSE`. Returns the status of a keyword execution to the server.
`BYE`	Client stops the test run. It should wait for `ACK`.
`ERR`	Client reports unrecoverable error in the adapter layer.
`ACK`	Positive acknowledgement: the last message (`BYE`) was understood.
`LOG`	Parameter required: free text without line breaks. The parameter is printed to the test log on the server side. This message can be sent whenever it is client's turn to talk.

	From server to client
`ACK`	Positive acknowledgement from the server: the last message from the client was understood. `ACK` is followed by a parameter (a keyword) if the last message was `GET`.
`NACK`	Negative response from the server: the last message was not understood.
`BYE`	Server stops the test run. Client should send `ACK` before disconnecting.
`ERR`	Server has detected an error (it found a difference between the expected and the observed behaviours). Client should repeat the last request.

3 Client implementation

A very simple client can be implemented as follows:

```
send HELO
receive ACK, otherwise quit with an error message

while true
    send GET
    receive a message
    if received BYE: send ACK and quit (test run ended)
    if received ERR: save information for debugging
    if received ACK:
        execute the keyword given as ACK parameter.
        repeat
            send PUT TRUE or PUT FALSE (status of the execution)
            receive a message
            if received BYE: send ACK and quit
            if received ERR: save information for debugging
        until received ACK
```

Clients can use the test log on the server side. Whenever it is client's turn to talk, it is allowed to send `LOG message`. The socket server adapter immediately writes `message` to the server log and sends `ACK`.

To interrupt a test run, a client sends `BYE`. To be sure that the server understood, the client should wait for `ACK`.

4 Communication example