htsh/wapsh – Implementation Notes

Copyright 2000, exolution GmbH / Michael Kerrisk, Munich, Germany
http://www.exolution.de/wapsh
mailto:wapsh@exolution.de

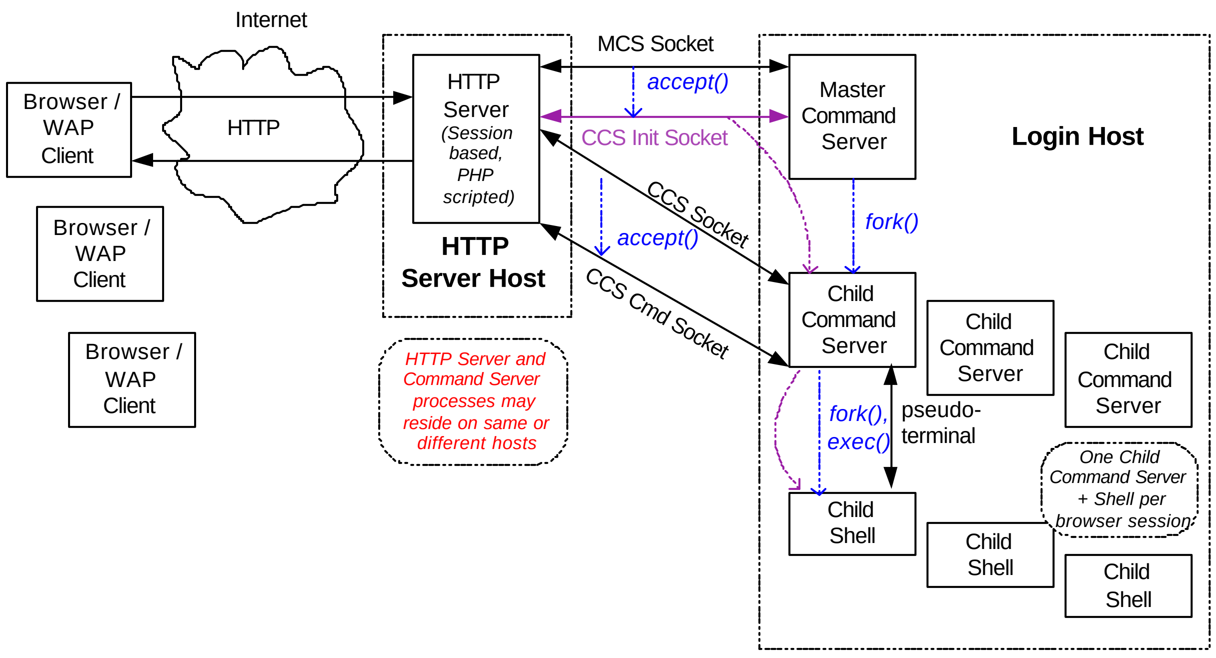
Last revised: 21 Nov 2000

This document provides (slightly thin!) background notes on the implementation of *htsh/wapsh*, a tool which allows a user operating a Web Browser or a WAP phone to operate a Unix shell on a remote machine. For an intrduction to *htsh* you should first read the *htsh/wapsh User Guide*.

1. Architecture Diagram

MCS = Master Command Server
CCS = Child Command Server
CSh = Child Shell

The term "Client Shell" refers to the pair of CCS and CSh processes which are dedicated to responding to a particular HTTP Session (logged in user).



The various socket connections shown in the above diagram have the following purposes:

- **MCS Socket** - Single instance, established on well known port. Accepts incoming connections for CCS initialisation socket.
- **CCS Initialisation Socket** - One instance per HTTP session, created by *accept()* from MCS socket, and inherited by CCS and Child Shell across *fork()s.* Used during initialisation of CCS, Child Shell and HTTP Session. Lifetime: until completion of initialisation
- **CCS Socket** - One instance per HTTP Session/CCS. Established on fixed port which is notified to HTTP Session. Lifetime: duration of HTTP Session / Client Shell.

- **CCS Command Socket** - Multiple instances per HTTP Session / CCS. Created by *accept()* on CCS Socket. Lifetime: time taken to read one shell input from HTTP session, pass to Child Shell, read any output, and pass back to HTTP Session.

Note that if *htshd* is run from *inetd(8)*, then there is no MCS process – instead each connection to the *htsh* services causes *inetd* to start a new *htshd* process which automatically configure itself as a CCS/CSh pair.

2. Initial Connection Establishment

1. The client browser sends a *username*, *password*, optional *login host*, and *client type* ("wap" or "http") via HTTP to the HTTP Server. (In the case of the WAP client this will involve an intervening WAP gateway).

2. The HTTP Server uses a socket connection to a well-known port, to send this information to the Master Command Server (MCS).

3. Upon receipt of a connection request, the MCS creates a Child Command Server (CCS) process which handles all further processing of the connection including the authentication of the user, and subsequent communication between the HTTP Server and Child Shell.

4. The CCS uses the `/etc/passwd` (and `/etc/shadow`) database to authenticate the user.

    a. If this fails an failure response is sent to the HTTP Server, which in turn informs the Browser Client

    b. If authentication succeeds:

        i. The CCS generates an authentication key which the HTTP Server must specify as part of all future messages.

        ii. The CCS parses the global and user-specific *htsh* initialisation files, obtaining information on *htsh* session options, and shortcut menu items.

        iii. The CCS creates a new (session) socket which will be used for communication between the HTTP Server (session) and the CCS. The port number of this socket is retrieved so that it can be notified to the HTTP Server which records it in Session data for this browser instance.

        iv. The CCS establishes a pseudo-terminal for communication with the Child Shell and *fork*/*execs* the Child Shell. From this point,

            1. The CCS operates a continuous loop transferring data in either direction between the HTTP Server and the Child Shell.

            2. The Child Shell (CSH):
                a. performs a range of initialisations, creating a *utmp* entry for this login session, setting the identity of the process, setting appropriate environment variables for the shell, and opening the lave end of the pseudo-terminal.
                b. Sends a notification message to the HTTP server, informing it of the socket port number for the Child Command Server, and the set of session options and shortcut commands defined in the *.htshrc* file.
                c. *Execs* the standard shell for this user

2.1. Authentication response message

One of the following messages is sent from the CCS/CSh at the completion of authentication/initialisation. Each of the lines in these messages is terminated by a null-byte (no terminating newline occurs).

2.1.1. Success message

A success message is indicated by a status code >= 0

status=*numeric code*
socketPort=*number*
authKey=*hex-number*
shellTimeout=*number*
setOptions=*octal-number*
numSc=*number*
sc0=*string*
sc1=*string*
…
scn=*string*
option-name=value
…

2.1.2. Failure message

A failure message is indicated by a status code < 0

status=*numeric code*
msg=*text*

3. Message exchange between HTTP server and Child Command Server (CCS)

On the HTTP server side, each message exchange between HTTP and Command Server involves:

1. Opening socket to Child Command Server
2. Sending one of the messages below to CCS
3. Closing socket

On the Child Command Server end:

1. (Block) Accept a new socket connection. (This is done with a timeout: if no new connection is received within the timeout period, the CCS and CSh exit)
2. Use socket to read message from HTTP server
3. Send appropriate string to child shell
4. Loop reading output from child shell, and passing to HTTP Server via connected socket. The loop reading from child shell is controlled by a *select()* with timeout (of some short period – say 0.5 secs). When no more input is received, or the limit specified by the *htshd –c* command line option is reached, we break out of loop.
5. Close connected socket.

3.1. Messages from HTTP to Child Command Server(CCS)

The following messages are sent from the HTTP server to the Child Command Server: All messages are sent in ASCII text. The first character of each message (after the hexadecimal string authorisation key) defines the command. Following this (for most messages) is a 4 digit string (present/absent according to the command type), and in the case of a shell command, a text string.

- **Shortcut: [authKey] [s] [# - shortcut number]**
  Ask remote shell to execute one of the Shortcut Commands (sc)

- **Shell command: [authKey] [c] [# - length of text][text]**
  Send textual input to shell. Text string should include newline character (if needed).

- **Close connection:** **[authKey] [z]**
  Closes connection, shuts down server

- **Control char:** **[authKey] [^] [char]**
  Send a control character to the remote shell. The character is specified using the corresponding printable ASCII characters (Thus ‘A’ (0x41) for Control-A (0x01)).

- **Signal:** **[authKey] [x] [# - signal-num]**
  Used to send numbered signal to remote shell (not currently used)

- **Null command:** **[authKey] [n]**
  Used to get any outstanding output from child shell

- **Set options:** **[authKey] [c] [# - length of text][text]**
  Specify a list of options to be set in the Child Command Server. These options take the form of a series of delimited NAME=VALUE strings. The length parameter of the command gives the length of the entire string of options.
  This feature is not as yet implemented

3.2. *Messages from Child Command Server to HTTP*

The following messages are sent from HTTP Server to Command Server Client. All messages are plain ASCII text. Each message begins with a single character indicating the message type. No length field is required as end of text is determined by end of file.

- **Normal text:** **[n] [text]**
  Normal textual output from remote child shell

- **Error condition** **[e] [text]**
  An error occurred on the CCS/CSh, as explained in *text*, however the CCS/CSh is still alive and prepared to accept further input.

- **Fatal error condition:** **[f] [text]**
  Some fatal error has occurred, the CCS/CSh is aborting, explanatory message in *text*.