

## A Complex Script, Explained

The following pages contain an example of a well-written and robust script.

This script uses most of DialScript's important features and is excessively commented in order to explain them.

```
-- Comments start with --.

-- To use 1200 bps, remove the "send break", and set speed to 1200
-- The script gets the modem's attention and dial, even if it has
-- to hang up to do it. It uses timeouts to recover from problems.
-- Problems after connect are assumed to be due to line noise, so it
-- hangs up and dials again in hope of a cleaner line. It assumes
-- a Hayes type modem set for English commands and responses.

-- You must set the variable for your username in state init and fix up
-- state FinishUp (optional).

script cs    -- scripts must begin with the word "script" and a name.
            -- Keywords like script must be in lower case.

            -- Execution begins with the first state in the file. In this case,
            -- the state named init. Every state must have a unique name.
            -- Identifiers (names) in DialScript are sequences of any
            -- length of letters, digits, and underscore characters. They
            -- must begin with a letter.

state init

            -- Display statements display characters on the terminal window
            -- without sending them. Use it for user messages. The variable
            -- date contains the current date and time. A newline is not
            -- automatically included. Hence the display "\r" at the end.

display "Beginning UT CS login script on\r";
display date;
display "\r"; -- <-- Don't forget semicolons after statements.

set port modem; -- The set statement is used to set communication
set speed 2400; -- parameters. These values are the defaults, so
set databits 8; -- these statements are not really necessary.

set online on; -- Be sure DialScript is online so that it will
              -- talk to the serial port. This is also
              -- the default.

            -- setvar is used to give variables values. All variables hold
            -- string values, never numbers, i.e. "67" not 67.
            -- Use variables to hold parameters that you may wish to change
            -- later.

setvar USERNAME "newton\r"; -- You must change this, of course.
setvar PHONE "ATDT4718454\r"; -- It's MY username, not yours.
setvar Modem_Escape_String "+++"; -- Some people change this.
```

```

-- input prompts the user for a value for a variable. The noecho
-- keyword causes what the user types to not be displayed. Good
-- for passwords.

input PASSWORD noecho; -- You could use a setvar here.

next "ModemReady"; -- Branch to the state named ModemReady.

end; -- init

-- This state makes sure we have the (Hayes-type) modem's attention.
-- It sends a carriage return to it and expects it to reply with OK.

state ModemReady

-- The repeat statement executes the statements before its
-- end a fixed number of times, twice in this case.

repeat 2

    -- send sends characters out over the serial port. Note that
    -- carriage return is not included automatically. Use
    -- \r for a carriage return.

    send "\r"; send "AT\r";

    -- select waits for one of several conditions.
    select
        "OK": next "Dial"; -- If OK is received, go to state Dial.
        timeout 3: -- If 3 seconds pass without a match,
            -- display a message and exit the select.
            display "ModemReady timeout!\r";
    end; -- select
end; -- repeat

-- If we get here, the select has timed out in both
-- iterations of the repeat. The modem is not responding.
-- Try hanging up.

next "HangUp"; -- failed again, maybe hangup

end; -- ModemReady

-- The state hangs up a Hayes modem by sending +++, waiting for OK,
-- and then sending ATH.

state HangUp

repeat 2
    -- Note that send pauses for one second before sending.
    -- The delay does nothing for 1 second to give an even greater
    -- pause before sending the escape string.
    delay 1; send Modem_Escape_String;
    select
        "OK" : send "ATH\r"; next "ModemReady";

```

```

        timeout 3 : display "HangUp timeout!\r";
    end;
end;

-- If we reach this point, we have not received the ack for the
-- escape string. We are confused and so try hanging up.
-- Control really should not reach this point.

send "\r"; send "ATH\r";
next "ModemReady";

end; -- HangUp

-- This state dials the phone number and awaits the CONNECT message.
-- The select causes a redial if the line is busy, the modem responds
-- with NO CARRIER, or the modem does not respond with 25 seconds.

state Dial
    send PHONE;    -- The system's phone number
    select
        "CONNECT"  : next "GotIt";
        "BUSY"     : next "ModemReady";
        "NO CARRIER" : next "ModemReady";
        timeout 25  : display "Dial timeout!\r"; next "ModemReady";
    end;
end; -- Dial

-- This state enters the username and the password in reponse to
-- the appropriate prompts. If there is no prompt within 60 seconds,
-- the script hangs up and redials.

state GotIt

    -- Here is a trick. The machine we are calling requires
    -- that we send a break in order to switch to 2400 baud and prompt
    -- for login. The delay 1 gives a little time between the modem
    -- connect and sending the break. It is probably not necessary.

    delay 1;
    send break;    -- UNIX host needs a break to switch to 2400 baud
    select
        "login:"   : send USERNAME;
        timeout 60 : display "login timeout!\r"; next "HangUp";
    end;

    select
        "Password:" : send PASSWORD; send "\r"; -- Your password
        timeout 60  : display "password timeout!\r"; next "HangUp";
    end;

    next "FinishUp";
end; -- GotIt

-- This state is used to answer the terminal type prompt. The nature
-- of this prompt depends on your .login file on UNIX. You need to
-- to customize this state for your circumstances. I enter return
-- to confirm that I will use a vt100 emulator and then switch to
-- the emulator I use (MacLayers) by means of the transfer.

```

```
-- "RunLayers" is my settings file for MacLayers. This script has  
-- to be in the same folder as MacLayers and RunLayers in order  
-- for the transfer to succeed. The transfer quits DialScript and  
-- runs MacLayers with settings file RunLayers (as though I  
-- had double clicked on RunLayers from the finder).
```

```
state FinishUp -- You need to customize this. What's here is weak.  
  -- wait "(vt100)"; -- Terminal type prompt  
  -- send "\r"; -- Yes to vt100  
  -- transfer "MacLayers" "RunLayers"; -- Run real term emulator  
end; -- FinishUp
```

```
-- Finally, the script ends with "end;"  
end; -- cs
```