DialScript 1.7

Manual by Peter Newton and Werner Uhrig

Distribution Notice!!

DialScript is distributed as **FreeWare** (which means that it is copyrighted by the author to retain certain rights over it). It is made available free of charge and may be distributed freely, as long as the following conditions are respected:

You may freely distribute the unmodified DialScript in both source and object format provided that you include this manual with it and that you do not charge for it. You may modify it anyway you please, but if you want to make it available thus to others, we require that you name this (your) program differently and make no references to DialScript or the author except, perhaps, in an acknowledgements section of the manual or the "About" box in the program. However, you may not sell or charge for a program that was developed based on DialScript.

DialScript is not warranted to do anything useful (it may well not for you!) and we disclaim all responsibility for any and all possible direct or indirect damages its use might cause. Use DialScript at your own risk.

Table of Contents

Purpose of DialScript

DialScript is intended to complement terminal emulation programs that lack a good macro capability for automatic and *intelligent*' login. Typically, users run DialScript to set up the modem, dial a phone number and step through a login sequence for a remote computer. When the login is complete, they switch either automatically or manually to a standard terminal emulator and proceed.

DialScript is an interpreter for an extremely simple programming language roughly based on finite state automata. Statements in the language interact with the serial ports by either sending strings to them or waiting for strings from them. The language also deals with time and timeouts. Using it, you can create quite powerful scripts that can redial when the line is busy and react to errors or exceptional conditions.

Instructions for Installing DialScript (This is all Apple's fault!)

If you are running system 7.0 or later, and your keyboard lacks a control key, you must drag file DialScript.Layout (without renaming it) onto your system file. This installs a keymapping resource that causes the option key to work as the control key while DialScript is running.

If you are running a system older than 7.0, you do not need to do anything special -- just run DialScript.

How to Use DialScript

When you run DialScript, a terminal window will appear. This window does not emulate any particular terminal type and is intended only to help you write scripts. Some communications parameters may be changed using the "Settings" menu. DialScript also contains a built-in editor (select either "New Edit" or "Open Edit" from the file menu). You may use this editor or any other TEXT file editor to create and view scripts. An example script that you could enter and run follows. More complex examples are distributed with DialScript (and you may find one explained in detail in the Appendix)

```
script Example1  -- This script dials a number on a Hayes modem
 state One
        send "AT\r";
        wait "OK";
        send "ATDT3456789\r";
 end;
end;
```

This script dials a number on a Hayes-type modem. Once it has been entered and saved, this script may be run by selecting "Run Script" from the File-menu. Press command-period to stop a running script. The DialScript language is case sensitive and contains lots of reserved words such as **script** and **state** (see "The DialScript Language" below).
The "\r" is used to represent a carriage return.

If you run a script containing a syntax error, DialScript will complain. Enable the Listing window (see the Settings-menu) and run the script again to obtain a script listing o identify the approximate location of the error.

Script files that were created by DialScript may be run by double clicking on them when DialScript is not running. This is a convenient way to run already finished scripts. Note, however, that DialScript will not automatically change the creator ID of files created by other programs; but you could use the Save As command to create a Dialscript document file or use the Kiss File feature to change a File's creator ID.

If you wish to log the text appearing in the Terminal window to a disk file for archival or later review, use the Log File feature (in the File menu). Old log files will be overwritten if you start a new one with the same name. Also, please note the **startlog** and **stoplog** statements below.

The DialScript Language

A brief description of the DialScript language follows. Looking at the example scripts supplied with DialScript may help also. In the description of the language, the following words have the meanings shown.

ident	An identifier, a sequence of any length of letters, digits, and underscores beginning with a letter.
number	A non-negative integer (stored in 32 bits).
string	A string constant.
value	Either a string or an identifier serving as a variable. All DialScript variables are global and of type string.

Strings are enclosed in double quotes and may contain some C-language "\" escapes for special characters -- '\\', '\"', '\r', '\n', '\t,' \'b,' '\f', and '\nnn' are recognized. The "nnn" in the last case represents up to three octal digits. Strings may not contain the Null character (\000).

Variable **date** is special. It contains the current date and time. You may not set it.

Each (named) *Script* consists of a sequence of uniquely named *States.* The *initial state* (i.e where execution begins) is the first one in the file. States consist of a sequence of statements that are executed sequentially, one after another, beginning with the first. *Keywords* (reserved) are displayed in **bold**. DialScript is case sensitive. Hence all keywords must be typed in lower case. Note the semicolons. The script and state forms (syntax) are as shown.

script ident
    <list of states>
end; **end**;

state ident
    <list of statements>

The possible statements follow.

setvar ident value;

Set variable ident to the given value. Setvar statements, like all statements, may appear only within state definitions.

input ident;

input ident value;
input ident **noecho**;

Prompt the user for a value for variable ident. Provide the default value, if present. The variable name will be part of the prompt. A carriage return is not included in the input value. The noecho form causes the input string not to be displayed as it is entered. You cannot use a default value with the noecho form.

send value;
send break;

Send a value, or a break after a one second delay. If a CarriageReturn is desired, it must be explicitly indicated with '**\r**'.

wait value;

Wait (indefinitely) for the value to be sent from the host. Characters displayed on the screen before the wait begins are not matched. It does not "look back". The results of waiting for the null string ("") are undefined.

display value;

Display a value on the Mac terminal window without sending it. Carriage return is not automatically included-- use \r. Display is useful to give status and progress information to the user.

delay number;

Do nothing for "number" seconds. Display but do not match all incoming data.

next ident;

Branch to the state named. State names must be unique (else the behavior might be unpredictable).

stop;

Terminate a script. Scripts also terminate upon reaching an **end**.

startlog value;

startlog value **append**;

Begin logging text from the terminal window to a file named *value*. The first form will overwrite an existing file of the same name; the second appends to it (if it does not exist, it creates it). Log files are created in the *current folder* (usually determined by using an OPEN-dialog in one of the File-menu commands - or the folder in which the currently running script is located)

stoplog;

Cease logging incoming text (and flush and close log file).

set speed number;	(Default: 2400)
set databits 7 | 8;	(Default: 8)
set stopbits 1 | 2;	(Default: 1)
set port printer | **modem**;	(Default: modem)
set parity none | **even** | **odd**;	(Default: none)
set dtr on | **off**;	(Default: on)
set online on | **off**;	(Default: on)

Set communication parameters. Supported speeds are 300, 1200, 2400, 4800, 9600, 19200, 38400, and 57600 Baud. **Set dtr** pulls the DTR-voltage high or low. Various modems react differently to this, and results can depend on your modem cable. The default is to pull DTR high. DialScript leaves DTR unchanged when it exits. So if it is high, it stays that way (this is important as some modems else might hang up when DialScript exits).

Understanding and programming the use of the serial port can be rather tricky when you want to have several communications programs take turns at exchanging data with another computer to which a link has been established on a given serial port (without losing the connection and without getting the programs and computers "all confused"). The **online** option controls whether or not the Mac's serial port is flagged as *currently-in-use* by DialScript. When the of state of **online** is **off**, DialScript *has closed* the port and will not attempt to send or receive data. Some other application may now open the serial port to communicate with the host, linked to by DialScript, without interference by Dialscript; under Multifinder it is not necessary to quit Dialscript. Be very careful never to have two applications use the same serial port at the same time. We are under the impression that most communications programs we have seen are *neither pleased* nor *prepared* to easily cooperate in sharing a serial port under any circumstance. The recent versions of VersaTerm do; MacLayers is currently being worked on and may be able to *go offline* already when you read this.

transfer value;
transfer value value;

Quit DialScript and run the named application (the first value), which will be instructed to open the document (second value), if it is supplied. The intended use is to chain to a terminal emulator program (such as MacLayers or ZTerm, etc.). Put DialScript, the application, and all associated documents in the same folder. Actually it suffices if the application you will transfer to, and any document you will attach are in the same folder with the Dialscript script you are making the transfer from. DialScript, the application file, can be somewhere else. If this confuses you, just put everything in the same folder.

select
value : <list of statements>
value : <list of statements>
*
*
timeout number : <list of statements>
end;

Wait for any of the values to match text sent from the host and then execute the corresponding statement list. If there is no match within "number" seconds, execute the timeout's statement list. There may be zero to 32 strings and zero or one timeout clause in a select. Do not try to match the null string ("").

repeat number
<list of statements>
end ;

Repeat a list of statements a fixed number of times. It is possible to branch out of a repeat using a next statement.

Acknowledgements

DialScript is based upon TransSkel by Paul DuBois and is implemented in THINK C 5.0.1 with the help of flex and bison from the GNU project. All of these packages are highly recommended. The latest version of DialScript, its source code, and all Freeware packages used are available via anonymous ftp to host rascal.ics.utexas.edu (128.83.138.20). Rascal does not have a mail server.

If you use DialScript, I'd be pleased to hear from you (to know the size of audience). Is it useful that I distribute this hack? Send me e-mail or a postcard, as might be most convenient for you.

Peter Newton
(newton@cs.utexas.edu or ...!uunet!cs.utexas.edu!newton)
8524 Burnet Rd #431
Austin, TX 78758
U.S.A.

Bugs and Hints

* Use select and timeout instead of wait to build robust scripts.
* Don't use timeout values that are too small. Humans tend to underestimate. Time it!
* Use display to print a message when a timeout occurs -- helps in debugging scripts.
* Error messages are vague and the listing window is slow.
* You may not put null characters or 8 bit characters into strings.
* Some script errors are not caught until runtime -- undefined state and too many select conditions are examples. Duplicate script names are not caught at all. The first is used. I am lazy.
* keys: OPTION is CONTROL, CLEAR is ESC, and SHIFT-BACKSPACE is DEL on MacPlus keyboards.
* Transfer is problematic. It works on System 6.0, but has a few problems with System 7.0. You may not refer to aliases in a transfer, and there *may* be problems transferring to applications that support the System 7.0 required Apple Events. MacLayers 1.0 and 1.10 do not and ZTerm 0.85 does not. The bottom line is try it and see if it works! I use it with MacLayers on System 7.0.
* The terminal window has no cursor.
* Does not run in the background under Multifinder. I would like it to, but it's too much of a hassle. This is Freeware, you know. Let's all bug Apple for preemptive multitasking like real computers have.
* DialScript operates with all flow control disabled.
* I intended DialScript to work on all Macs from the 512 up, using System 4.2 or later. Does it? I don't know. During the development of DialScript I have used it on Plus, SE, SE/30, IIcx, and a Classic under systems 6.0.x and under 7.0.
* Option key remapping is always active while DialScript is running under Finder (as opposed to Multifinder). This can cause the option key to act incorrectly within desk accessories that are running with DialScript.
* Using the printer port while Appletalk is active causes a crash.
* Has USA keyboard dependencies. This problem might go away if you run System 7.0 and do NOT install the keyboard layout. This will make sense only for Macs without control keys. Please tell me if this does or does not work. I have no way to test it. For other circumstances, I believe that those who know how can hack the internal KCHR (for System 6.0.x) or the layout file (for System 7.0) with ResEdit to resolve the keyboard dependencies. Again, please let me know.

* The trace features in the listing dialog are useless. Ignore them.
* If you use Appletalk Remote, you must turn of "Answer Calls" in order to effectively use DialScript. If you do not, the Appletalk Remote software will hang up your modem whenever the serial port is closed. Hence, you will be disconnected when you switch from DialScript to your terminal emulator.

A Complex Script Explained

The following pages contain an example of a well-written and robust script that may be used to login to machine cs.utexas.edu at the University of Texas at Austin. (If you have an account on it, of course.) The machine runs a variant of BSD 4.3 UNIX. If you are at the University of Texas, check the Mac archives on rascal.ics.utexas.edu by anonymous ftp. There are some scripts for other UT machines there.

This script uses most of DialScript's important features and is excessively commented in order to explain them. It script may also be found in file cs.ds in the example folder distributed with DialScript.

```
-- UTexas CS Dept. 2400 Baud login script.            Vers. 11/4/91

-- To use 1200 bps, remove the "send break", and set speed to 1200

-- The script tries to get 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 you have
-- a Hayes type modem set for English commands and responses.

-- You need to 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 will begin 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

     -- The display statement displays characters on the Mac terminal window
     -- without sending them.  Use it for informative messages.  The variable
     -- date contains the current date and time.  A newline is not auto-
     -- matically included.  Hence the display "\r" at the end.

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

     set port modem;  -- The set statements is used to set communications
     set speed 2400;  -- parameters.  These values are the defaults, so these
     set databits 8;  -- 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 a value to a variable.  All variables hold
     -- string values, never numbers, i.e. "67" not 67.  It is convenient
```

```
   -- to use variables to hold parameters that you may wish to change
   -- later.

   setvar USERNAME "newton\r";   -- You need to change this, of course.
   setvar PHONE "ATDT4718454\r"; -- It's MY username, not yours.
   setvar Modem_Escape_String "+++";  -- Some people like to 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 repeats 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 it.
      send "\r"; send "AT\r";

      -- select waits for one of several conditions.
      select
        "OK":          -- If OK is received, go to state Dial.
           next Dial;
        timeout 3:     -- If 3 seconds pass without a condition matching,
                       -- display a warning message and leave 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 we have a bit of a trick.  The machine we are calling requires
    -- that we send a break.  It will then 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 and return
      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 terminal emulator

  end;  -- FinishUp


  -- FInally, the script ends with "end;"

end; -- cs
```