

# DialScript 1.6

Copyright 1990, 1991 Peter Newton

## Notice

DialScript may be distributed without charge in both source and object form provided you include this manual with it. However, you may not sell it or minor modifications of it. You may, however, include it with a product you sell. If you modify and redistribute DialScript, please choose a new name for your program and do not include my name on it except perhaps in an acknowledgements section of the manual or "about" box. I do not warrant that DialScript will do anything useful (it may well not!), and I disclaim responsibility for any direct or indirect damages caused by its use. Use DialScript at your own risk.

## Purpose of DialScript

DialScript is intended to complement (not replace) terminal emulation programs that lack an adequate automatic login facility. Typically, users run DialScript to dial a phone number and perform a login sequence for a remote computer. When the login is complete, they switch either automatically or manually to a real terminal emulator and proceed.

DialScript is an interpreter for an extremely simple programming language 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.

## 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. Communications parameters may be changed via 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.

```

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. The "\r" represents carriage return. 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.

If you try to run a script that has a syntax error, DialScript will complain. Enable the Listing window (Settings menu) and run it again to obtain a script listing that shows 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 change the creator ID of files created by other programs.

## The DialScript Language

The language is simple enough that you may learn it by looking at the examples supplied, but the following brief summary may help also.

A script consists of a name and a sequence of states, each with a unique name. The initial state (first to be run) is the first one in the file. States consist of a sequence of statements that are executed one after another beginning with the first. Keywords (reserved) are shown in bold. DialScript is case sensitive so they must be typed in lower case. Note the semicolons. The script and state forms are as shown.

<pre> <b>script</b> name     &lt;list of states&gt; <b>end;</b> </pre>	<pre> <b>state</b> uniqueName     &lt;list of statements&gt; </pre>
--	---

There are many different statement types in the DialScript language. They are listed below.

**setvar** varname "string";

Set variable varname to a string value. All variables are global and take only string values. The main use of variables is in conjunction with input to avoid putting passwords into files where others may see them.

**input** varname; or **input** varname "default value string";

Prompt the user for a string value for variable varname. Provide the default value, if present. The variable name will be part of the prompt. A carriage return will not be included in the input value;

**send** "string"; or **send** varname; or **send break**;

Send the string shown, the contents of a variable, or a break after a one second delay. Carriage return (\r) is not automatically included.

**wait** "string";

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

**display** "string";

Display the string on the Mac terminal window without sending. Carriage return and newline are automatically included.

**delay** number;

Do nothing for "number" seconds.

**next** statename;

Branch to the state named.

**stop**;

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

<b>set speed</b> number;	(Default: 2400)
<b>set databits</b> 7   8;	(Default: 8)
<b>set stopbits</b> 1   2;	(Default: 1)
<b>set port printer</b>   <b>modem</b> ;	(Default: modem)
<b>set parity</b> none   even   odd;	(Default: none)
<b>set dtr</b> high   low;	(Default: high)

Set communication parameters. Supported speeds are 300, 1200, 2400, 4800, 9600, 19200, 38400, and 57600 Baud. Set DTR pulls the DTR pin 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 to prevent some modems from hanging up.

**transfer** "application name"; or  
**transfer** "application name" "document";

Quit DialScript and run the named application, which will be instructed to open the document, if it is supplied. Use this to chain to a terminal emulator. Put DialScript, the application, and all associated documents in the same folder. (Actually, you can put the script you will run, the application you will transfer to, and any document you will attach in the same folder. DialScript can be somewhere else. If this confuses you, just put everything in the same folder.)

```
select
  "string1" : <list of statements>
  "string2" : <list of statements>
  *
  *
  timeout number : <list of statements>
end;
```

Waits for any of the strings to match text sent from the host and then executes 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 ("").

Identifiers (names) are sequences of letters, digits, and underscores that start with a letter. They may be any length. Numbers are sequences of digits. They must be nonnegative, and they are stored in 32 bits. Strings are enclosed in double quotes and may contain the C language "\" escapes for special characters-- '\', '\'', '\r', '\n', '\t', '\b', '\f', and '\nnn' are provided. The "nnn" in the last case represents up to three octal digits. Strings may not contain a null, however.

### 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.

- \* Command-period stops scripts. I have heard that this may not work on some types of non-U.S.A. keyboards. It's tough being foreign! (I'd try to fix this, but I do not know how.)
- \* Option is control, clear is esc, and shift backspace is del on MacPlus keyboards.
- \* Transfer (especially with attached document) is based on code that Apple warns may not work on future systems.
- \* 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 intend DialScript to work on all Macs from the 512 up under system 4.2 or later. Does it? I don't know. I've tested various versions of it on a plus, an SE, an SE/30, a IIcx, and a classic under system 6.0.x and on a plus under 7.0.
- \* Option key remapping is always active while DialScript is running under finder (as opposed to mulifinder). This can cause the option key to act incorrectly when desk accessories are running with DialScript.
- \* Using the printer port while Appletalk is active causes a crash.

### Acknowledgements

DialScript is based upon TransSkel by Paul DuBois and is implemented in THINK C 5.0 with the help of flex and bison from the GNU project. All of these packages are highly recommended. DialScript source code is available via anonymous ftp to rascal.ics.utexas.edu (128.83.138.20).

If you want, send me e-mail or a postcard if you use DialScript. I'd like to know the size of audience. Is it useful that I distribute this hack?

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