DialScript Language Reference

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), e.g. 17.

string A string constant, e.g. "hello".

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 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 one after another beginning with the first. Keywords are reserved-- you cannot use them as identifiers. DialScript is case sensitive. Hence all keywords must be typed in lower case. Note the semicolons that end statements. The script and state forms (syntax) are as shown. Keywords are in bold type.

```
script ident                    state ident
    <list of states>                <list of statements>
end ;                                                   end ;
```

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 out over the serial line after a one second delay. If a carriage return is desired, it must be explicitly indicated with '\r'.

```
wait value ;
```

Wait (indefinitely) for the value to be received 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. DialScript's text matching is case-sensitive.

```
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 characters.

```
next value ;
```

Branch to the state with name given by the value. State names must be unique. Since the argument is a value, a direct reference to a state name should be in double quotes. An argument that is not in quotes is a variable name. However, if such a variable is not set to a value, DialScript interprets the name as a statename. In other words,

```
next foo ;
```

branches to the state with name stored in variable foo, unless nothing is stored in foo in which case control passes to the state named foo. This bizzare semantics is designed to allow scripts written for prior versions of DialScript to run under the current version.

```
stop ;
```

Terminate a script. Scripts also terminate upon reaching the end of a state.

```
startlog value ;
startlog value append ;
```

Begin logging text from the terminal window to a file whose name is given by 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 the file buffer and close the 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 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.