Dial-Up Scripting Command Language
For Dial-Up Networking Scripting Support

Table of Contents

1.0 Overview

Many Internet service providers and online services require you to manually enter information, such as your user name and password, to establish a connection. With Scripting support for Dial-Up Networking, you can write a script to automate this process.

A script is a text file that contains a series of commands, parameters, and expressions required by your Internet service provider or online service to establish the connection and use the service. You can use any text editor, such as Microsoft Notepad, to create a script file. Once you've created your script file, you can then assign it to a specific Dial-Up Networking connection by running the Dial-Up Scripting Tool.

2.0 Basic Structure of a Script

A command is the basic instruction that a script file contains. Some commands require parameters that further define what the command should do. An expression is a combination of operators and arguments that create a result. Expressions can be used as values in any command. Examples of expressions include arithmetic, relational comparisons, and string concatenations.

The basic form of a script for Dial-Up Networking follows:

```
;
; A comment begins with a semi-colon and extends to
; the end of the line.
;

proc main
      ;  A script can have any number of variables
```

```
        ;  and commands

        variable declarations

        command block

endproc
```

A script must have a main procedure, specified by the **proc** keyword, and a matching **endproc** keyword, indicating the end of the procedure.

You must declare variables before you add commands. The first command in the main procedure is executed, and then any subsequent commands are executed in the order they appear in the script. The script ends when the end of the main procedure is reached.

3.0 Variables

Scripts may contain variables. Variable names must begin with a letter or an underscore ('_'), and may contain any sequence of upper- or lower-case letters, digits, and underscores. You cannot use a reserved word as a variable name. For more information, see the list of reserved words at the end of this document.

You must declare variables before you use them. When you declare a variable, you must also define its type. A variable of a certain type may only contain values of that same type. The following three types of variables are supported:

Type	Description
integer	A negative or positive number, such as 7, -12, or 5698.
string	A series of characters enclosed in double-quotes; for example, "Hello world!" or "Enter password:".
boolean	A logical boolean value of TRUE or FALSE.

Variables are assigned values using the following assignment statement:

variable **=** *expression*

The variable gets the evaluated expression.

Examples:

```
integer count = 5
integer timeout = (4 * 3)
integer i

boolean bDone = FALSE

string szIP = "getip 2"

set ipaddr szIP
```

3.1 System Variables

System variables are set by scripting commands or are determined by the information your enter when you set up a Dial-Up Networking connection. System variables are read-only, which means they cannot be changed within the script. The system variables are:

Name	Type	Description
$USERID	String	The user identification for the current connection. This variable is the value of the user name specified in the Dial-Up Networking Connect To dialog box.
$PASSWORD	String	The password for the current connection. This variable is the value of the user name specified in the Dial-Up Networking Connect To dialog box.
$SUCCESS	Boolean	This variable is set by certain commands to indicate whether or not the command succeeded. A script can make decisions based upon the value of this variable.
$FAILURE	Boolean	This variable is set by certain commands to indicate whether or not the command failed. A script can make decisions based upon the value of this variable.

These variables may be used wherever an expression of a similar type is used. For example,

```
transmit $USERID
```

is a valid command because $USERID is a variable of type string.

4.0 String Literals

Scripting for Dial-Up Networking supports escape sequences and caret translations, as described below.

String Literal	Description
^*char*	Caret translation If *char* is a value between '@' and '_', the character sequence is translated into a single-byte value between 0 and 31. For example, ^M is converted to a carriage return. If *char* is a value between a and z, the character sequence is translated into a single-byte value between 1 and 26. If *char* is any other value, the character sequence is not specially treated.
<cr>	Carriage return
<lf>	Linefeed
\"	Double-quote
\^	Single caret
\<	Single '<'
****	Backslash

Examples:

```
transmit "^M"
transmit "Joe^M"
transmit "<cr><lf>"
waitfor "<cr><lf>"
```

5.0 Expressions

An expression is a combination of operators and arguments that evaluates to a result. Expressions can be used as values in any command.

An expression can combine any variable, or integer, string, or boolean values with any of the unary and binary operators in the following tables. All unary operators take the highest precedence. The precedence of binary operators is indicated by their position in the table.

The unary operators are:

Operator	Type of Operation
-	Unary minus
!	One's complement

The binary operators are listed in the following table in their order of precedence. Operators with higher precedence are listed first:

Operators	Type of Operation	Type Restrictions
* /	Multiplicative	Integers
+ -	Additiveintegers	Strings (+ only)
< > <= >=	Relational	Integers
== !=	Equality	Integers, strings, booleans
and	Logical AND	Booleans
or	Logical OR	Booleans

Examples:

```
count = 3 + 5 * 40
transmit "Hello" + " there"
delay 24 / (7 - 1)
```

6.0 Comments

All text on a line following a semicolon is ignored.

Examples:

```
; this is a comment

transmit "hello"          ; transmit the string "hello"
```

7.0 Keywords

Keywords specify the structure of the script. Unlike commands, they do not perform an action. The keywords are listed below.

proc *name*

Indicates the beginning of a procedure. All scripts must have a main procedure (**proc** main). Script execution starts at the main procedure and terminates at the end of the main procedure.

endproc

Indicates the end of a procedure. When the script is executed to the **endproc** statement for the main procedure, Dial-Up Networking will start PPP or SLIP.

integer *name* [**=** *value*]

Declares a variable of type integer. You can use any numerical expression or variable to initialize the variable.

string *name* [**=** *value*]

Declares a variable of type string. You can use any string literal or variable to initialize the variable.

boolean *name* [**=** *value*]

Declares a variable of type boolean. You can use any boolean expression or variable to initialize the variable.

8.0 Commands

All commands are reserved words, which means you cannot declare variables that have the same names as the commands. The commands are listed below:

delay *nSeconds*

Pauses for the number of seconds specified by *nSeconds* before executing the next command in the script.

Examples:

```
delay 2      ; pauses for 2 seconds
delay x * 3  ; pauses for x * 3 seconds
```

getip *value*

Waits for an IP address to be received from the remote computer. If your Internet service provider returns several IP addresses in a string, use the *value* parameter to specify which IP address the script should use.

Examples:

```
; get the second IP address
set ipaddr getip 2
```

```
; assign the first received IP address to a variable
szAddress = getip
```

goto *label*

Jumps to the location in the script specified by *label* and continues executing the commands following it.

Example:

```
  waitfor "Prompt>" until 10
  if !$SUCCESS then
       goto BailOut ; jumps to BailOut and executes commands
                    ; following it
  endif

  transmit "bbs^M"
  goto End

BailOut:
  transmit "^M"
```

halt

Stops the script. This command does not remove the terminal dialog window. You must click Continue to establish the connection. You cannot restart the script.

if *condition* **then**
commands
endif

Executes the series of *commands* if *condition* is TRUE.

Example:

```
if $USERID == "John" then
       transmit "Johnny^M"
endif
```

label **:**

Specifies the place in the script to jump to. A label must be a unique name and follow the naming conventions of variables.

set port databits 5 | 6 | 7 | 8

Changes the number of bits in the bytes that are transmitted and received during the session. The number of bits can be between 5 and 8. If you do not include this command, Dial-Up Networking will use the properties settings specified for the connection.

Example:

```
set port databits 7
```

set port parity none | odd | even | mark | space

Changes the parity scheme for the port during the session. If you do not include this command, Dial-Up Networking will use the properties settings specified for the connection.

Example:

```
set port parity even
```

set port stopbits 1 | 2

Changes the number of stop bits for the port during the session. This number can be either 1 or 2. If you do not include this command, Dial-Up Networking uses the properties settings specified for the connection.

Example:

```
set port stopbits 2
```

set screen keyboard on | off

Enables or disables keyboard input in the scripting terminal window.

Example:

```
set screen keyboard on
```

set ipaddr *string*

Specifies the IP address of the workstation for the session. *String* must be in the form of an IP address.

Examples:

```
szIPAddress = "11.543.23.13"
set ipaddr szIPAddress

set ipaddr "11.543.23.13"

set ipaddr getip
```

transmit *string* [**, raw**]

Sends the characters specified by *string* to the remote computer.

The remote computer will recognize escape sequences and caret translations, unless you include the **raw** parameter with the command. The **raw** parameter is useful when transmitting $USERID and $PASSWORD system variables when the user name or

password contains character sequences that, without the **raw** parameter, would be interpreted as caret or escape sequences.

Examples:

```
transmit "slip" + "^M"
transmit $USERID, raw
```

waitfor *string* [**, matchcase**] [**then** *label*
{ **,** *string* [**, matchcase**] **then** *label* }]
[**until** *time*]

Waits until your computer receives one or more of the specified strings from the remote computer. The *string* parameter is case-insensitive, unless you include the **matchcase** parameter.

If a matching string is received and the **then** *label* parameter is used, this command will jump to the place in the script file designated by *label*.

The optional **until** *time* parameter defines the maximum number of seconds that your computer will wait to receive the string before it execute the next command. Without this parameter, your computer will wait forever.

If your computer receives one of the specified strings, the system variable $SUCCESS is set to TRUE. Otherwise, it is set to FALSE if the number of seconds specified by *time* elapses before the string is received.

Examples:

```
waitfor "Login:"

waitfor "Password?", matchcase

waitfor "prompt>" until 10

waitfor
       "Login:"     then DoLogin,
       "Password:"  then DoPassword,
       "BBS:"       then DoBBS,
       "Other:"     then DoOther
  until 10
```

while *condition* **do**
commands
endwhile

Executes the series of *commands* until *condition* is FALSE.

Example:

```
integer count = 0

while count < 4 do
       transmit "^M"
       waitfor "Login:" until 10
       if $SUCCESS then
```

```
                goto DoLogin
            endif
            count = count + 1
        endwhile
        ...
```

9.0 Reserved Words

The following words are reserved and may not be used as variable names.

and	boolean	databits	delay	
do	endif	endproc		endwhile
even	FALSE	getip	goto	
halt	if	integer	ipaddr	
keyboard	mark	matchcase	none	
odd	off	on	or	
parity	port	proc	raw	
screen	set	space	stopbits	
string	then	transmit	TRUE	
until	waitfor	while		