Table of Contents

2. Introduction

Script files are simple programs in the QWinsock script language. This language is similar to a simplified version of the BASIC programming language with some additions to support login requirements of various Service Providers.

NOTE: This document is not intended to provide instruction in programming fundamentals and concepts. Its intent is to provide a brief overview of the Qwinsock script language and reference to its keywords and commands. If you are not familiar with programming concepts, it will be helpful if you refer to a book on BASIC programming.

3. Identifiers (names)

Identifiers are used to represent variables, labels, and commands. Identifiers can be any combination of alphanumeric characters and the underscore character and are not case sensitive. For example, the identifier "myname" is the same as "MyName" and "MYNAME."

The script language reserves a number of names for its language definition. These *internal commands* can not be used as Variable names. These reserved names include the following:

abort	goto	then
destroy	if	to
do	next	trace
else	number	until
end	print	wend
endif	repeat	while
for	return	
gosub	string	

A number of commands, also called *run time commands*, may not be used as Variable names in login scripts. These reserved command names include the following:

CfgSetValue	CommWaitFor	GetInput
CfgGetValue	CommReadWord	GetPassword
CommSend	CommReadIPAddr	SetTimeout

3.1. Labels

Labels specify a point in the script program that can by used as the target of a GOTO or GOSUB command. A Label begins with a colon and is followed by a series of alpha or numeric characters. Labels must appear in the first column of a line.

The following are examples of valid Labels:

:Label
:500
:ErrorRoutine

Labels are not case sensitive. As a result, the following two Label definitions are equivalent:

:LabelA
:labela

Defining two Labels with the same or equivalent name will cause an error.

3.2. Strings

Constant strings are made up of printable characters enclosed within double quotes. The length of a string is limited to 65535 bytes or your editor's line limit, which ever is less. You can also include the punctuation and non-printing characters listed in the table below.

%r	carriage return
%n	line feed
%t	tab
%b	backspace
%"	double quote
%%	percent sign (%)

3.3. Numbers

Constant numbers are integer values comprised of one or more digits . The integer range of Numbers is -2,147,483,648 to 2,147,483,647.

3.4. Variables

Two types of variables are supported by the script language, *String* and *Number*. Variables are defined by specifying the variable type (String or Number) followed by the name to assign the variable. In addition, a variable may be assigned the value of a constant or previously defined variable.

The following are examples of valid Variable definitions:

String lastname
String lastname = "Smith" *creates a String variable called "lastname" and assigns the value "Smith" to the variable.*

Number index
Number counter = 0 *creates a Number variable called "counter" and assigns the value "0" to the variable.*

Number savecounter = counter *assigns the value of the variable "counter" to the second variable "savecounter"*

Hint: As a reminder of how variable assignment works, read the "=" sign as "gets the value of." In the Number variable example, you would read the line as "Counter gets the value of" zero."

Type conversion between Strings and Numbers is performed automatically by the script engine. Assignment of a Number to a String results in an alpha representation of the number. For example, this would convert the number "11" to a string comprised of two 1's. The reverse assignment, String to Number, results in a conversion of the string value to its closest integer value supported by the range of Number. If a type conversion from a String to a Number can not be performed, the destination Number is assigned a value of zero.

Variable names are not case sensitive. As a result, the following variable definitions all define a variable with the same name:

```
String MyName
String myname
Number myName
Number MYNAME
```

Defining two variables with the same or equivalent names will cause an error. The example above, which creates four equivalent variables, would cause an error.

Hint: Variables and Labels are represented independently within the script engine. As a result, a Variable may have the same name as a Label. The following example shows a completely valid Label and Variable definition:

```
:Age
Number Age
String Name
:Name
```

3.5. Predefined Variables

The script engine predefines the Number variable RESULT for use by run time commands to set a result code. This result code may be a return value or an indication of an error. In other words, the run time commands

assign a value to the variable RESULT, which you can test to see whether the command was successful.

Note: only the *run-time* commands modify the RESULT variable (see the second table on page 2); the *internal* commands do not. Each run-time variable uses the RESULT variable in its own way; be sure to see the documentation for these run time commands to determine their use, if any, of the RESULT variable.

4. Command Reference

The next section is a reference guide for the available Winsock script commands. Some conventions used in the reference guide:

- Commands are commented in italics. That is, next to each line of Winsock script commands is a short explanation in italic type; the italicized words are not part of the command. For example:

Command	Explanation
End	*this command ends the program*

- The vertical bar in a command's "Syntax" separates a list of possible command options from which you should select *one*. For example:

  Syntax: Trace ON | OFF

  This means that the command line should read "Trace ON" or "Trace OFF" — but **not** "Trace" (alone) or "Trace ON OFF".

4.1. Internal Commands

4.1.1. Program Termination Commands

4.1.1.1. End

Description Ends execution of the script program and indicates to the calling program that the script completed successfully.

Syntax End

Example End

See Also Abort

4.1.1.2. Abort

Description Ends execution of the script program and indicates to the calling program that the script did not complete successfully. The ABORT command may optionally display an error message to the output window using the same parameter format as the PRINT command.

Syntax Abort [*String* | *stringvariable*]

Example

```
String Username

CfgGetValue "Username" username  gets the username from the config file

if RESULT = 0 then                    if 0 (no user name) then abort with an
   Abort "Error reading username"     error message
   endif
```

See Also End, Print

4.1.2. Branching Commands

The following commands allow branching to a predefined label.

4.1.2.1. Gosub

Description Provides an unconditional jump to a label with a return of control to the subsequent command. Control is passed back to the command following the Gosub command via the Return command. The Gosub command may be nested.

Syntax Gosub *label*

Example

```
String password                       define the String variable "password"

CfgGetValue "Password" password  get the password from the config file

if RESULT = 0 then                    if 0 (no password) then jump to the
  Gosub PromptForPassword             label PromptForPassword
  endif

:PromptForPassword                    label ":PromptForPassword"

UserInput "Password" password      ask the user to enter a password

if RESULT = 0 then                    if 0 (user did not enter one) abort with
  Abort "Password required to log in" error message
 endif
Return                                Return to the line after the Gosub
                                      command
```

See Also Return, Goto (Use Goto if you want to jump to a label but do not want to return to the same place.)

4.1.2.2. Goto

Description Provides an unconditional jump to a label.

Syntax Goto *label*

Example

```
String Username                          define string variables
String Password

CfgGetValue “Username” Usernameget the user name from the config file
if RESULT = 0 then                       if 0 (no user name) then jump to the
label
  Goto ReadError                                        ”ReadError”
  endif

CfgGetValue “Password” password  get the password from the config file
if RESULT = 0 then                       if 0 (no password) then jump to the label
  Goto ReadError                                        “ReadError”
  endif
.
.
:ReadError                               label “ReadError”
Abort “Error reading username or password”   abort with error message
```

See Also Gosub (Use Gosub when you want to jump to a label but soon thereafter to return to the same place.)

4.1.2.3. Return

Description Terminates the GOSUB commands by passing control (that is, “returning”) to the command immediately following the GOSUB command. If the GOSUB command is not terminated with a RETURN, the script is terminated with an error.

Syntax

```
Return
```

Example

```
Gosub ReadPassword                       jump to the label “ReadPassword”
.
.
.
:ReadPassword                            label

String Password                          define the string variable “Password”

CfgGetValue “Password” Password  get the password from the config file

if RESULT = 0 then                       if 0 (no password) then abort with an
error
  Abort “Error reading password”     message
  endif
Return                                   return to the line after the Gosub
command
```

See Also Gosub

4.1.3. Conditional Execution

The following commands allow the execution a group of commands or expressions to based on a the result of an expression resolving to TRUE. That is, if the test condition (such as "is this value equal to 2?") is true, the conditional commands will execute.

4.1.3.1. If

Description Allows the execution of one or more statements to be dependent upon a specified condition. The IF command ***must*** be terminated by the ENDIF command. IF commands may be nested.

Syntax

```
If expression Then
    one or more commands or expressions
[Else]
    one or more commands or expressions
Endif
```

Example

```
String IPAddress                        define the string variable "IPAddress"

CommWaitfor "IP address = "             wait until the COM port reads "IP
                                        address"

CommReadIPAddress IPAddress     read the IP address from the COM port

if RESULT = 0 then                      if 0 (IP address did not get read)
  Abort "Error reading IP address from host"   abort with an error message
else                                    otherwise....
   CfgSetValue "IPAddress" IPAddress    Set the IP address in config file print an
   Print "IP address set to", IPAddress informative message
    endif                               end the if block
End                                     end the script
```

See Also Then, Else, Endif

4.1.3.2. Then

Description The THEN command indicates the begining of the command(s) to execute when the *expression* portion of the IF command resolves to TRUE. An IF command without a THEN command will cause an error.

See Also If, Else, Endif (For a notated example, see the **If** command on page 8.)

4.1.3.3. Else

Description Provides an alternate path of execution when the expression portion of the IF command resolves to FALSE. The ELSE command must precede the ENDIF command.

See Also If, Then, Endif (For a notated example, see the **If** command on page 8.)

4.1.3.4. Endif

Description Terminates the IF command statement. Each IF command block **must** end with the Endif command. If the IF command is not terminated with an ENDIF command, the script is terminated with an error.

See Also If, Then, Else (For a notated example, see the **If** command on page 8.)

4.1.4. Iterative Execution

The following commands provide methods for executing a group of commands or expressions multiple times.

4.1.4.1. Repeat

Description Allows a group of commands to be executed until a condition resolves to TRUE. The group of commands will be executed *at least one time.* The Repeat command is terminated by the Until command. Repeat commands may be nested.

Syntax

Repeat
one or more commands or expressions
Until *expression*

See Also Until, While

4.1.4.2. Until

Description Terminates the Repeat command statement. If the Repeat command is not terminated with the Until command, the script is terminated with an error.

See Also Repeat

4.1.4.3. While

Description Allows a group of commands to be executed while a condition is TRUE.

The While command differs from the Repeat command in that the condition is evaluated *before* the group of commands executed. As a result, the condition must be TRUE *before* any commands are executed.

The While command's condition must be followed by the **Do** command. The While command is terminated by the **Wend** command. While commands may be nested.

Syntax

While *expression* Do
one or more commands or expressions
Wend

Hint: To make a command repeat indefinitely, use the expression "While 1 = 1".

See Also Do, Wend, Repeat

4.1.4.4. Do

Description The Do command indicates the begining of the command(s) to execute if the While expression resolves to TRUE. A While command without a Do command will cause an error.

See Also While, Wend

4.1.4.5. Wend

Description Terminates the While command statement. All commands between the Do and Wend commands will get executed if the While expression resolves to TRUE.

If the While command is not terminated with the Wend command, the script is terminated with an error.

See Also While, Do

4.1.4.6. For

Repeats a group of commands a fixed number of times. The FOR command uses a Number variable as a counter to keep track of the current iteration and requires a *starting value* to initialize the counter and an *ending value* to compare against the counter to determine if the correct number of iterations have been performed.

FOR *counter* = *start* TO *end*
one or more commands or expressions
NEXT

counter	*Previously defined* Number variable used as a loop counter
start	Number constant used as the initial value of counter
TO	keyword separating *start* and *end*
end	Number constant used as the final value of counter

Example

```
Number I                          define the counter variable

For I = 1 to 10                   repeat this ten times
Print "This will print ten times." print a message
Next                              ends the for...next block
```

4.1.4.7. To

Keyword used to separate the *start* and *end* values of the FOR command.

4.1.4.8. Next

Ends the FOR loop and causes the *counter* to be incremented by one. All commands between the FOR and NEXT command are considered. If the FOR command is not terminated with a NEXT command, the script is terminated with an error.

4.1.5. Variable Commands

The following commands allow creating and destroying variables.

4.1.5.1. String

Creates a named variable for storing alphanumeric text.

Syntax:

STRING *variablename*

4.1.5.2. Number

Creates a named variable for storing integer numbers.

Syntax:

NUMBER *variablename*

4.1.5.3. Destroy

Destroyes a previously defined String or Number variable.

Syntax:

DESTROY *variablename*

4.1.6. Display Commands

4.1.6.1. Print

The PRINT command allows you to display quoted strings (such as “Press Return please”) as well as Constant or Variable identifiers. When displaying multiple Constant or Variable identifier, each must be separated by a comma or a semicolon. The *semicolon* will insert a blank between the two values and the *comma* will insert a Tab between the two values.

PRINT *identifier* [, | ; *identifier*] [, | ; *identifier*] ... [, | ; *identifier*]

Examples

Print command	Prints as
Print “Good morning.”	Good morning.
String User = “John.”	
Print “Good morning,”, User	Good morning, John

See the table in the section on Strings on page 3 if you need to insert non-printing characters, quotes or percent signs in a PRINT command.

4.1.6.2. Trace

The TRACE command causes all commands or expressions executed to be displayed to the output window as their are executed.

Syntax: TRACE ON | OFF

4.2. External "Run Time" Commands

4.2.1. Configuration Commands

4.2.1.1. CfgSetValue

Description The CfgSetValue allows saving a value for the provider to the INI file. This command is used for setting values read from the COM port such as the IP address, DNS server, etc.

Syntax CfgSetValue *variablename* | *String identifier2*

Example

```
String IPAddress                        define the variable "IPAddress"

CommWaitfor "ipaddress ="               listen to the COM port for the string
                                        "ipaddress ="

CommReadIPAddress IPAddress             read from the COM port into the variable
                                        IPAddress

if RESULT = -1 then                     if -1 (problem reading the address)
  Abort "Error reading IP address from host"  abort with a message
  endif                                 end the "if" block
```

***NOTE**: If the program gets this far, there was no error reading from the COM port.*

```
CfgSetValue "IPAdddress" IPAddress      set the IPAddress config file directive
with
                                        the contents of the variable "IPAddress"
```

4.2.1.2. CfgGetValue

Description The CfgGetValue reads a provider value from the INI file. This command is used for reading values such as the Username or Password for later tranmission out the COM port.

Syntax CfgGetValue *variablename* | *String stringvariable*

RESULT variable The length of the data retreived from the INI file. A result of 0 means that no data was read from the INI file.

Example

```
String Uname                          declare some string variables
String Pword

CfgGetValue "Username" Uname          get the Username information from the
                                      config file and assign it to the variable
                                      "username"

if RESULT = 0 then                    if 0 (problem) then
   Abort "User name not set"          abort with an error message
   endif                              end the "if" block

otherwise....

CfgGetValue "Password" Pword          get the config file password information
                                      and assign it to the variable "Pword"

if RESULT = 0 then                    if 0 (problem) then
   Abort "Password not set"           abort with an error message
   endif                              end the "if" block
```

4.2.2. COM Read/Write commands

4.2.2.1. CommSend

Description The CommSend commands allows sending a variable or constant data to the COM port.

Syntax CommSend *variablename* | *String* | *Number*

RESULT variable The RESULT variable will be set to the 0 if all data was sent or -1 if an error occured.

Example

```
String Uname                          declare a variable

CfgGetValue "Username" Uname          get the Username parameter from the
                                      config file

if  RESULT = 0 then                   if 0 (problem getting the parameter)
   Abort "Username not set"           abort with an error message
   endif                              end the "if" block

                                      otherwise...

CommWaitfor "login:"                  listen at the COM port for the string
                                      "login:"

CommSend Uname                                send the contents of
the variable "Uname"
                                      through the COM port
```

```
if RESULT = -1 then                    if -1 (problem sending) then
   Abort "Error sending Username"      abort with an error message
   endif                               end the "if" block
```

4.2.2.2. CommWaitFor

Description The CommWaitFor command allows the script to pause until a specific character string has been received on the COM port. All data up to and including the wait string is discarded.

Syntax CommWaitFor *variablename* | *String*

Example

```
String Uname                           declare a variable

CfgGetValue "Username", Uname          get the Username value from the config
                                       file and assign it to the variable
"Uname"

CommWaitFor "login:"                   listen at the COM port for the string
                                       "login:"

CommSend Uname                         send the contents of
the variable"Uname"
                                       through the COM port
```

4.2.2.3. CommReadIPAddr

Description The CommReadIPAddr command allows reading the next word from the COM port as an IP Address.

Syntax CommReadIPAddr *stringvariable*

RESULT variable The length of the IP address string on success and zero on failure.

Example

```
String ipaddress                       declare a variable to hold the IP
address

CommReadIPAddr ipaddress               read an IP address from the COM port

if RESULT > 0 then                     if the length of the string just read
CfgSetValue "IPAddress" ipaddress      exceeds 0, set the IPAddress config file
                                       parameter to the contents of the
variable
                                       "ipaddress"
endif                                  end the "if" block
```

4.2.3. Miscellaneous commands

4.2.3.1. SetTimeout

Description The SetTimeout command resets the timeout for establishing the connection. This command is used in cases where the default ConnectTime setting may be too short to allow the script to complete successfully. The value specified is the number of seconds to continue connecting before timing out.

Syntax SetTimeout *Number* | *numbervariable*

Example SetTimeout 60

4.2.3.2. GetInput

Description This command is used to prompt the user for input and store the result in a variable

Syntax GetInput "promptstring" variable

Note that the prompt string must be in double quotes.

RESULT variable The length of the text typed by the user.

Example

```
String Uname                          declare a variable

GetInput "Enter your name" Uname ask the user to enter a username and store
                                      the user's input in the variable"Uname"

if RESULT > 0 then                    if the length of the text just entered
                                    exceeds 0 then
Print Uname                           print the contents of the variable
"Uname"
  endif                               end the "if" block
```

4.2.3.3. GetPassword

Description This command is used to prompt the user for input and store the result in a variable. This function is identical to the GetInput command with the exception that the input field in the dialog box displays an astericks for each character typed instead of the actual character

Syntax GetPassword "promptstring" variable

Note that the prompt string must be in double quotes.

RESULT variable The length of the text typed by the user.

Example

```
String Uname                                    declare a variable

GetPassword “Enter your password” Uname         ask the user to enter a username and
store
                                                the user’s input in the variable“Uname”

if RESULT > 0 then                              if the length of the text just entered
exceeds
                                                0, then
Print Uname                                     print the contents of the variable
"Uname”
   endif                                        end the “if” block
```

4.2.3.4. Delay

Description The Delay command introduces a pause in the execution of the script for a specified number of seconds.

Syntax Delay *seconds*.