

Quick3270TM

Macro Language Reference Manual

Version 4.31 – February 2012



74, rue Verte
34980 Saint-Gély-du-Fesc – France
Phone: +33 675 369 257

Email: contact@dn-computing.com
www: <http://www.dn-computing.com/>

Table of CONTENTS

Introduction	1
Variables and Data Types	2
Data Types	2
Variable identifiers	2
Label identifiers	2
Declaring Variable.....	2
Dim Statement.....	3
Const Statement.....	3
Control Commands	4
Call Statement	4
End	4
Exit.....	4
For...Next Statement.....	4
GoSub	5
GoTo.....	5
If...Then...Else Statement.....	6
Quit	6
Return.....	7
Stop.....	7
Sub...End Sub Statement	7
While...Wend Statement.....	8
Whith...End With Statement.....	8
String functions	10
Chr or Chr\$.....	10
InStr	10
LCase or LCase\$	11
Len	11
Mid	11
Space or Space\$.....	12
Str or Str\$	12
StrComp.....	13
UCase or UCase\$	13
Val.....	14
File functions and statements.....	15
Open.....	15
Close.....	15
LineInput or Line Input.....	16
Print #	16
Write #	17
Seek #	17
Get #	18
Put #	18
EOF.....	18
FreeFile	19
Loc	19

LOF	19
Kill	19
COM functions	20
CreateObject	20
GetObject	21
Miscellaneous functions and statements	22
Beep	22
Date\$	22
Decrypt or Decrypt \$	22
Encrypt or Encrypt\$	23
Environ or Environ\$	23
GetOpenFileName	24
InputDialog	24
MsgBox Function	25
PasswordBox	27
Rem	27
Shell	28
Time\$	28
Is	29
IsNull	29
VarType.....	29
Quick3270 functions and statements	30
EditCopy	30
EditCopyAdd	30
EditPaste	30
GetRow	31
GetCol	31
GetColor	31
GetIpAddress	32
GetMacroKeystroke	33
GetOIA	34
GetString	34
GetTextRect	35
GetTimeoutValue	35
GetVisible	35
GetWindowTitle	36
MoveTo	36
MoveRelative.....	36
OpenPrinter	37
ClosePrinter	37
PrintScreenFontSize	37
PrintScreen	38
PutString	38
ReceiveFile.....	39
Search	40
SendDarkKeys.....	40
SendFile	41
SendKeys	42
SetText	44
SetTimeoutValue	44
SetVisible	45

SetWindowTitle	45
Updated	45
WaitForAttrib.....	46
WaitForCursor	47
WaitForCursorMove.....	47
WaitForKbdUnlock.....	48
WaitForString	48
WaitHostQuiet	49

Functions supported in IBM® Personal Communications™ compatibility mode . 50

autECLSession.SetConnectionByName	50
autECLSession.autECLOIA.WaitForAppAvailable	51
autECLSession.autECLOIA.WaitForInputReady	51
autECLSession.autECLXfer.ReceiveFile.....	52
autECLSession.autECLXfer.SendFile	53
autECLSession.autECLPS.CursorPosRow	54
autECLSession.autECLPS.CursorPosCol	54
autECLSession.autECLPS.GetTextRect	54
autECLSession.autECLPS.SetText.....	55
autECLSession.autECLPS.SendKeys.....	55
autECLSession.autECLPS.Wait	58
autECLSession.autECLPS.WaitForAttrib	59
autECLSession.autECLPS.WaitForCursor	60
autECLSession.autECLPS.WaitForString	60
autSystem.Inputnd	61

Functions supported in Attachmate’s Extra!™ compatibility mode..... 62

Macro Samples 63

Select an Excel file using GetOpenFileName	63
Creating a Reference to an Excel Object	63
File I/O.....	63
Connecting to a data source and executing SQL Queries	64
Find Values from screen and use the WScript object.....	64
Simple logon script	64
Using LDAP to retrieve the user name and password and start logon script	65
PComm VBScript sample supported by Quick3270	66
PComm Macro sample supported by Quick3270	67
Attachmate’s Extra! macro sample supported by Quick3270.....	68

Introduction

A macro is a series of recorded or coded commands you can run to complete tasks more quickly. Macros allow you to condense complex Quick3270 commands into a single command or keystroke.

There are two ways to create a macro with Quick3270. You can record the keystrokes you enter as you're working, and/or you can use the Macro Editor in conjunction with the Quick3270 Macro Language to code your macro.

To automate a series of tasks.

Instead of entering your user ID, password, and pressing the ENTER key each time you log on to your mainframe account, you can automate the process with a macro.

To automate an often-used sequence of commands and actions.

If you routinely open a mainframe application as part of your log on sequence, you can create a macro to open the application for you. If you've worked with macros or done some Basic programming before, much of the language information will be familiar to you.

Record a Keystroke Sequence

If you regularly do the same things when you work with a host system, it is convenient to record the keystrokes you make and have Quick3270 play them back if you want to do the same job again. All your keystrokes can be saved to a file; when you play the file back (Run), everything that happened will be reproduced.

Macro Language

The macro language allows you to create scripts for use with Quick3270. Presented here are some script functions that are recognized by Quick3270. Creating or modifying macros should be done by those who have programming experience and are knowledgeable in VBScript.

Variables and Data Types

In Quick3270 Basic, you use variables and constants to store values. Variables can contain data represented by any supported data type.

Data Types

The following table lists the fundamental data types that the Macro language supports.

Data type	Description	Range
Integer	4-byte integer	– 2,147,483,648 to 2,147,483,647.
String	String of characters	Zero to 65535 characters
Boolean	2 bytes	True or False
Object	4 bytes	Any object reference.

Note: Quick3270 language doesn't support arrays and floating point variables (Single / Double)

Variable identifiers

The first character must be an alphabetic (A-Z, a-z) or an underscore character "_". Subsequent characters can be alphabetic, underscore or numeric (0-9). Variable identifiers are not case sensitive. The maximum length is 32.

Label identifiers

Label identifiers consist of alphabetic, underscore or numeric characters, and are not case sensitive. The maximum length is 32.

```
goto label
```

```
...
```

```
label:
```

Declaring Variable

A variable is created the first time a value is assigned to it, as shown in the following samples.

```
MyVar = 459
```

```
MyString = "Hello world"
```

```
MyObject = CreateObject("Excel.Application")
```

Dim Statement

Description

Declares variables and allocates storage space

Syntax

Dim *varname* **As** type

```
Dim MyVar As Integer
Dim MyString As String
Dim MyObject As Object
```

Const Statement.

Description

Declares constants for use in place of literal values.

Syntax

Const *constname* = *expression*

```
Const MyVar = 459
Const MyString = "Hello world"
```

Control Commands

Call Statement

Description

Transfers control to another macro file.

When the **Exit** command in the second macro file is encountered, execution returns to the command that follows **Call** statement

Up to 10 levels of macro files can be called

Syntax

Call *MacroFilename*

End

Description

Terminates an application.

Syntax

End

Exit

Description

Immediately exits the execution of the current macro file and returns to the calling macro file if any.

Syntax

Exit

For...Next Statement

Description

Repeats a group of statements a specified number of times.

Syntax

For *counter* = *start* **To** *end* [**Step** *step*]
 [*statements*]

Next

The **For...Next** statement syntax has these parts:

Part	Description
counter	Numeric variable used as a loop counter.
start	Initial value of counter.
end	Final value of counter.
step	Amount counter is changed each time through the loop. If not specified, step defaults to one.
statements	One or more statements between For and Next that are executed the specified number of times.

GoSub

Description

Subroutine call. When you use the **GoSub** command, execution is transferred to the specified label and continues until a **Return/End Sub** command is encountered, at which point execution returns to the command that follows **GoSub**.

Syntax

GoSub *label*

or

GoSub *Subroutine name*

where *label* is defined by the **Label** identifier and Subroutine name defined by a **Sub** procedure.

In case of a Sub procedure, the **GoSub** statement is optional. A Sub procedure can be called directly by its procedure name.

```
GosubDemo ' or.Gosub GosubDemo
```

```
Sub GosubDemo()
```

```
...
```

```
End Sub ' following the GoSub statement.
```

Remark

Added in version 3.95

GoTo

Description

Branches unconditionally to a specified label.

Syntax

GoTo *label*

where *label* is defined by the Label command.

If...Then...Else Statement

Description

Conditionally executes a group of statements, depending on the value of an expression.

Syntax

If *condition* **Then** *statements* [**Else** *elstatements*]

Or, you can use the block form syntax:

```
If condition Then  
    [statements]  
[ElseIf condition-n Then  
    [elseifstatements]] . . .  
[Else  
    [elstatements]]  
Endif
```

The **If...Then...Else** statement syntax has these parts:

Part	Description
condition	One or more of the following two types of expressions: A numeric or string expression that evaluates to True or False. If condition is Null, condition is treated as False.
statements	One or more statements separated by colons; executed if condition is True.
condition-n	Same as condition.
elseifstatements	One or more statements executed if the associated condition-n is True.
elstatements	One or more statements executed if no previous condition or condition-n expression is True.

Remarks

You can use the single-line form (first syntax) for short, simple tests. However, the block form (second syntax) provides more structure and flexibility than the single-line form and is usually easier to read, maintain, and debug.

Quit

Description

Terminates a macro (same as Exit).

Syntax

Quit

Return

Description

Returns to the command that follows the last executed **GoSub** command.

Syntax

Return

Stop

Description

Terminates a macro (same as Exit).

Stop statements can be placed anywhere in a program to suspend its execution.

Syntax

Stop

Remark

Added in version 3.95

Sub...End Sub Statement

Description

Executes a series of statements within the body of the Sub procedure.

Syntax

Sub *name*
 [*statements*]
End Sub

The Sub...End Sub statement syntax has these parts:

Part	Description
name	Name of the Sub; follows standard variable naming conventions..
statements	Any group of statements to be executed within the body of the Sub procedure.

Remarks

If condition is **True**, all statements in statements are executed until the **Wend** statement is encountered. Control then returns to the **While** statement and condition is again checked. If condition is still **True**, the process is repeated. If it is not **True**, execution resumes with the statement following the **Wend** statement.

While...Wend loops can be nested to any level. Each **Wend** matches the most recent **While**.

Statement added in version 3.95

While...Wend Statement

Description

Executes a series of statements as long as a given condition is True.

Syntax

While *condition*

Version [*statements*]

Wend

The While...Wend statement syntax has these parts:

Part	Description
condition	Numeric or string expression that evaluates to True or False. If condition is Null, condition is treated as False.
statements	One or more statements executed while condition is True.

Remarks

If condition is **True**, all statements in statements are executed until the **Wend** statement is encountered. Control then returns to the **While** statement and condition is again checked. If condition is still **True**, the process is repeated. If it is not **True**, execution resumes with the statement following the Wend statement.

While...Wend loops can be nested to any level. Each **Wend** matches the most recent **While**.

Whith...End With Statement

Description

Executes a series of statements making repeated reference to a single object or structure.

Syntax

Whith *object*

[*statements*]

End With

The While...Wend statement syntax has these parts:

Part	Description
object	Required. Variable or expression. Can evaluate to any data type, including elementary types.
statements	Optional. One or more statements between With and End With that run on object.

Remarks

Added in version 4.31

With...End With allows you to perform a series of statements on a specified object without requalifying the name of the object.

Nesting Structures. Quick3270 don't allow to nest **With...End With** structures.

String functions

Chr or Chr\$

Description

Returns the character associated with the specified ANSI character code.

Syntax

Chr(*charcode*)

The *charcode* argument is a number that identifies a character

Remarks

Numbers from 0 to 31 are the same as standard, nonprintable ASCII codes. For example, **Chr**(10) returns a linefeed character.

The following example uses the **Chr** function to return the character associated with the specified character code:

```
Dim MyChar
MyChar = Chr(65)      ' Returns A.
MyChar = Chr(97)      ' Returns a.
MyChar = Chr(62)      ' Returns >.
MyChar = Chr(37)      ' Returns %.
```

InStr

Description

Returns the position of the first occurrence of one string within another.

Syntax

InStr(*string1*, *string2*)

The InStr function syntax has these arguments:

Part	Description
string1	Required. String expression being searched.
string2	Required. String expression searched for.

Return Values

The **InStr** function returns the following values:

If	InStr returns
string2 is not found	0
string2 is found within string1	Position at which match is found

LCase or LCase\$

Description

Returns a string converted to lowercase.

Syntax

LCase(*string*) or **LCase\$**(*string*)

Len

Description

Returns the number of characters in a string.

Syntax

Len(*string*)

Mid

Description

Returns a specified number of characters from a string.

Syntax

Mid(*string*, *start*[, *length*])

The **Mid** function syntax has these arguments:

Part		Description
string	String	String expression from which characters are returned.
start	Integer	Character position in string at which the part to be taken begins..
length	Integer	Number of characters to return. If omitted or if there are fewer than length characters in the text (including the character at start), all characters from the start position to the end of the string are returned.

Space or Space\$

Description

Returns a string consisting of the specified number of spaces.

Syntax

Space(*number*) or **Space\$**(*number*)

The number argument is the number of spaces you want in the string.

Remarks

The following example uses the Space function to return a string consisting of a specified number of spaces:

```
Dim MyString
MyString = Space(10) ' Returns a string with 10 spaces.
MyString = "Hello" & Space(10) & "World" ' Insert 10 spaces between two
strings.
```

Str or Str\$

Description

Returns a String representation of a number.

Syntax

Str(*number*)

The required number argument is a Integer containing any valid numeric expression.

Remarks

When numbers are converted to strings, a leading space is always reserved for the sign of number. If number is positive, the returned string contains a leading space and the plus sign is implied.

StrComp

Description

Returns a value indicating the result of a string comparison.

Syntax

StrComp(*string1*, *string2*)

The **StrComp** function syntax has these arguments:

Part	Description
string1	Required. Any valid string expression.
string2	Required. Any valid string expression.

Return Values

The **StrComp** function has the following return values:

If	StrComp returns
string1 is less than string2	-1
string1 is equal to string2	0
string1 is greater than string2	1

UCase or UCase\$

Description

Returns a string converted to uppercase.

Syntax

UCase(*string*) or **UCase\$**(*string*)

Val

Description

Returns the numbers contained in a string as a numeric value of appropriate type.

Syntax

Val(*string*)

The required string argument is any valid string expression.

Remarks

The **Val** function stops reading the string at the first character it can't recognize as part of a number. Symbols and characters that are often considered parts of numeric values, such as dollar signs and commas, are not recognized. However, the function recognizes the radix &H (for hexadecimal).

File functions and statements

Open

Description

Enables input/output (I/O) to a file.

Syntax

Open *pathname* **For mode** **As** [#]*filenumber* [**Len**=*reclength*]

The Open statement syntax has these parts:

Part	Type	Description
Pathname	String	Required. String expression that specifies a file name - may include directory or folder, and drive.
Mode		Required. Keyword specifying the file mode: Append, Binary, Input, Output, or Random. If unspecified, the file is opened for Random access.
Filenumber	Integer	Required. A valid file number in the range 1 to 10, inclusive. Use the FreeFile function to obtain the next available file number.
Reclength	Integer	Optional. Number less than or equal to 32,767 (bytes). For files opened for random access, this value is the record length. For sequential files, this value is the number of characters buffered.

Remarks

You must open a file before any I/O operation can be performed on it. Open allocates a buffer for I/O to the file and determines the mode of access to use with the buffer.

Close

Description

Concludes input/output (I/O) to a file opened using the **Open** statement.

Syntax

Close *filenumber*

The *filenumber* is any valid file number:

Remarks

When the **Close** statement is executed, the association of a file with its file number ends.

LineInput or Line Input

Description

Reads a single line from an open sequential file and assigns it to a String variable.

Syntax

LineInput #*filename*, *varname*

The **LineInput** # statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
varname	String	Required. Valid String variable name.

Remarks

The **LineInput** # statement reads from a file one character at a time until it encounters a carriage return (Chr(13)) or carriage return–linefeed (Chr(13) + Chr(10)) sequence.

Print #

Description

Writes display-formatted data to a sequential file.

Syntax

Print #*filename*, [*outputlist*]

The **Print** # statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
outputlist*		Optional. Expression or list of expressions to print.

Remarks

Data written with **Print #** is usually read from a file with **LineInput #**.

outputlist* Quick3270 supports only one expression. Not possible currently to specify a list of expressions delimited by commas.

Write #

Description

Writes data to a sequential file.

Syntax

Write #*filename*, [*outputlist*]

The **Write #** statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
outputlist		Optional. One or more comma-delimited numeric expressions or string expressions to write to a file.

Seek #

Description

Sets the position for the next read/write operation within a file opened using the Open statement.

Syntax

Seek #*filename*, *position*

The **Seek #** statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
position	Integer	Required. Number in the range 1 – 2,147,483,647, inclusive, that indicates where the next read/write operation should occur.

Get #

Description

Reads data from an open disk file into a variable.

Syntax

Get # *filename*, [*recnumber*], *varname*

The **Get #** statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
recnumber	Integer	Optional. Byte number at which reading begins
varname	String	Required. Valid variable name into which data is read.

Put #

Description

Writes data from a variable to a disk file.

Syntax

Put # *filename*, [*recnumber*], *varname*

The **Put #** statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
recnumber	Integer	Optional. Byte number at which reading begins
varname	String	Required. Valid variable name into which data is read.

EOF

Description

Returns the Boolean value True when the end of a file has been reached.

Syntax

EOF(*filename*)

The required filename argument is an Integer containing any valid file number.

FreeFile

Description

Returns an Integer representing the next file number available for use by the Open statement.

Syntax

FreeFile()

Remarks

Use FreeFile to supply a file number that is not already in use.

Loc

Description

Returns an Integer specifying the current read/write position within an open file.

Syntax

Loc(*filenumber*)

The required filenumber argument is an Integer containing any valid file number.

LOF

Description

Returns an Integer representing the size, in bytes, of a file opened using the Open statement.

Syntax

LOF(*filenumber*)

The required filenumber argument is an Integer containing any valid file number.

Kill

Description

Deletes a file from a disk.

Syntax

Kill *pathname*

The required pathname argument is a string expression that specifies one a file name to be deleted. The pathname may include the directory or folder, and the drive.

COM functions

In Quick3270 Macro, you can access an object and use the originating software application to change properties and methods of that object. You can programmatically manipulate the data in these objects.

Before you can use an object in a procedure, however, you must access the software application associated with the object by assigning it to an object variable. Next, you attach an object name (with or without properties and methods) to the variable to manipulate the object.

CreateObject

Description

Creates and returns a reference to an Automation object.

Syntax

CreateObject(*servername.typename*)

The CreateObject function syntax has these parts:

Part	Type	Description
servername	String	Required. The name of the application providing the object.
typename	String	Required. The type or class of the object to create.

Remarks

Automation servers provide at least one type of object. For example, a word-processing application may provide an application object, a document object, and a toolbar object. To create an Automation object, assign the object returned by **CreateObject** to an object variable:

```
Set ExcelSheet = CreateObject("Excel.Sheet")
```

GetObject

Description

Returns a reference to an Automation object from a file.

Syntax

GetObject([*pathname*] [, *class*])

The **GetObject** function syntax has these parts:

Part	Type	Description
pathname	String	Optional. Full path and name of the file containing the object to retrieve. If pathname is omitted, class is required.
class	String	Optional. Class of the object.

The class argument uses the syntax appname.objecttype and has these parts:

Part	Type	Description
appname	String	Required. Name of the application providing the object.
objecttype	String	Required. Type or class of object to create.

Remarks

Use the **GetObject** function to access an Automation object from a file and assign the object to an object variable. Use the Set statement to assign the object returned by **GetObject** to the object variable. For example:

```
Set CADObject = GetObject("C:\CAD\SCHEMA.CAD")
```

When this code is executed, the application associated with the specified pathname is started and the object in the specified file is activated. If pathname is a zero-length string (""), **GetObject** returns a new object instance of the specified type. If the pathname argument is omitted, **GetObject** returns a currently active object of the specified type. If no object of the specified type exists, an error occurs.

Miscellaneous functions and statements

Beep

Description

Sounds a tone through the computer's speaker. The frequency and duration of the beep depends on hardware, which may vary among computers

Syntax

Beep

Date\$

Description

Returns a string containing the current system date.

Syntax

Date\$()

Decrypt or Decrypt \$

Description

Returns the String decrypted.

Syntax

Decrypt (str) or **Decrypt \$**(str)

The **Decrypt** function syntax has this part:

Part	Type	Description
str	String	Required; String expression that will be decrypted.

Remark

Added in version 4.30

Encrypt or Encrypt\$

Description

Returns the String encrypted.

Syntax

Encrypt(str) or **Encrypt \$**(str)

The **Encrypt** function syntax has this part:

Part	Type	Description
str	String	Required; String expression that will be encrypted.

Remark

Added in version 4.30

Environ or Environ\$

Description

Returns the String associated with an operating system environment variable.

Syntax

Environ(str) or **Environ\$**(str)

The **Environ** function syntax has this part:

Part	Type	Description
str	String	Required; String expression containing the name of an environment variable.

Remark

Added in version 4.30

GetOpenFileName

Description

This function creates a system-defined dialog box that enables the user to select a file to open.

Syntax

GetOpenFileName(*title*, *pathname* , *filter*)

The **GetOpenFileName** function syntax has these parts:

Part	Type	Description
title	String	Required. String to be placed in the title bar of the dialog box
pathname	String	Required. String that can specify the initial directory.
filter	String	Required. String containing pairs filters separated by the ' ' character.

InputDialog

Description

Displays a prompt in a dialog box, waits for the user to input text or click a button, and returns the contents of the text box.

Syntax

InputDialog(*prompt*[, *title*][, *default*])

The **InputDialog** function syntax has these parts:

Part	Type	Description
prompt	String	Required. String expression displayed as the message in the dialog box.
title	String	Optional. String expression displayed in the title bar of the dialog box. If you omit title, the application name is placed in the title bar.
default	String	Optional. String expression displayed in the text box as the default response if no other input is provided. If you omit default, the text box is displayed empty.

Remarks

If the user clicks OK or presses ENTER, the **InputDialog** function returns whatever is in the text box. If the user clicks Cancel, the function returns a zero-length string ("").

MsgBox Function

Description

Displays a message in a dialog box, waits for the user to click a button, and returns a value indicating which button the user clicked.

Syntax

MsgBox(*prompt*[, *buttons*][, *title*])

The **MsgBox** function syntax has these arguments:

Part	Type	Description
prompt	String	String expression displayed as the message in the dialog box. The maximum length of prompt is approximately 1024 characters, depending on the width of the characters used. If prompt consists of more than one line, you can separate the lines using a carriage return character (Chr(13)), a linefeed character (Chr(10)), or carriage return–linefeed character combination (Chr(13) & Chr(10)) between each line.
buttons	Integer	Optional. Numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. See Settings section for values. If omitted, the default value for buttons is 0.
title	String	String expression displayed in the title bar of the dialog box. If you omit title, the application name is placed in the title bar.

MsgBox Constants

The following constants are used with the MsgBox function to identify what buttons and icons appear on a message box and which button is the default. In addition, the modality of the **MsgBox** can be specified. Since these constants are built into Quick3270 Scripting language, you don't have to define them before using them. Use them anywhere in your code to represent the values shown for each.

Constant	Value	Description
vbOKOnly	0	Display OK button only.
vbOKCancel	1	Display OK and Cancel buttons.
vbAbortRetryIgnore	2	Display Abort, Retry, and Ignore buttons.
vbYesNoCancel	3	Display Yes, No, and Cancel buttons.
vbYesNo	4	Display Yes and No buttons.
vbRetryCancel	5	Display Retry and Cancel buttons.
vbCritical	16	Display Critical Message icon.
vbQuestion	32	Display Warning Query icon.
vbExclamation	48	Display Warning Message icon.
vbInformation	64	Display Information Message icon.

Constant	Value	Description
vbDefaultButton1	0	First button is the default.
vbDefaultButton2	256	Second button is the default.
vbDefaultButton3	512	Third button is the default.
vbDefaultButton4	768	Fourth button is the default.
vbApplicationModal	0	Application modal. The user must respond to the message box before continuing work in the current application.
vbSystemModal	4096	System modal. This constant provides an application modal message box that always remains on top of any other programs you may have running.
vbMsgBoxHelpButton	16384	Adds Help button to the message box
vbMsgBoxSetForeground	65536	Specifies the message box window as the foreground window
vbMsgBoxRight	524288	Text is right aligned
vbMsgBoxRtlReading	1048576	Specifies text should appear as right-to-left reading on Hebrew and Arabic systems

MsgBox Return Values.

The following constants are used with the **MsgBox** function to identify which button a user has selected. These constants are built into Quick3270 Scripting language; you don't have to define them before using them.

Constant	Value	Description
vbOK	1	OK button was clicked.
vbCancel	2	Cancel button was clicked.
vbAbort	3	Abort button was clicked.
vbRetry	4	Retry button was clicked.
vbIgnore	5	Ignore button was clicked.
vbYes	6	Yes button was clicked.
vbNo	7	No button was clicked.

PasswordBox

Description

Displays a prompt in a dialog box, waits for the user to input text or click a button, and returns the contents of the text box. The Input text is hidden and replaced by asterisk '*'.

Syntax

PasswordBox(*prompt*[, *title*][, *default*])

The **PasswordBox** function syntax has these parts:

Part	Type	Description
prompt	String	Required; String expression displayed as the message in the dialog box.
title	String	Optional; String expression displayed in the title bar of the dialog box. If you omit title, the application name is placed in the title bar.
default	String	String expression displayed in the text box as the default response if no other input is provided. If you omit default, the text box is displayed empty.

Remarks

If the user clicks OK or presses ENTER, the PasswordBox function returns whatever is in the text box. If the user clicks Cancel, the function returns a zero-length string ("").

Rem

Description

Used to include explanatory remarks in a program. You can use an apostrophe (') instead of the Rem keyword.

Syntax

Rem *Comment*

You can also use the following syntax:
' comment

Shell

Description

Runs an executable program and returns an Integer representing the program's task ID if successful, otherwise it returns zero.

Syntax

Shell(*pathname*[,*windowstyle*])

The **Shell** function syntax has these parts:

Part	Type	Description
pathname	String	Required; Name of the program to execute and any required arguments or command-line switches; may include directory or folder and drive.
windowstyle	Integer	Optional. Integer corresponding to the style of the window in which the program is to be run. If windowstyle is omitted, the program is started with focus.

The windowstyle named argument has these values:

Constant	Value	Description
vbHide	0	Window is hidden and focus is passed to the hidden window.
vbNormalFocus	1	Window has focus and is restored to its original size and position.
vbMinimizedFocus	2	Window is displayed as an icon with focus.
vbMaximizedFocus	3	Window is maximized with focus.
vbNormalNoFocus	4	Window is restored to its most recent size and position. The currently active window remains active.
vbMinimizedNoFocus	6	Window is displayed as an icon. The currently active window remains active.

Time\$

Description

Returns a string containing the current system time.

Syntax

Time\$()

Is

Description

Compares two object expressions and returns a value that indicates whether the expressions refer to the same object.

*If (expression1 **Is** expression2)*

Remarks

Added in version 3.96.

Also use the **Is** operator to test whether an object variable has been **Set** to **Nothing**.

IsNull

Description

Returns a Boolean value that indicates whether an expression contains no valid data (Null).

IsNull(*expression*)

The *expression* argument can be any expression.

Remarks

IsNull returns **True** if expression is Null, that is, it contains no valid data; otherwise, **IsNull** returns **False**. If expression consists of more than one variable, Null in any constituent variable causes True to be returned for the entire expression.

VarType

Description

Returns a value indicating the subtype of a variable.

VarType(*VarName*)

The *varname* argument can be any variable.

Return Values

The **VarType** function returns the following values:

Constant	Value	Description
vbEmpty	0	Empty (uninitialized)
vbInteger	3	Integer
vbString	8	String
vbObject	9	Automation object
vbBoolean	11	Boolean

Quick3270 functions and statements

EditCopy

Description

Copies the content of the selected presentation space area to the clipboard. If no area is selected, the entire presentation space is copied to the clipboard.

Remark

Added in version 3.94

Syntax

EditCopy

EditCopyAdd

Description

Append the content of the selected presentation space area to the clipboard. If no area is selected, the entire presentation space is appended to the clipboard.

Remark

Added in version 3.94

Syntax

EditCopyAdd

EditPaste

Description

Copies the content of the clipboard to the presentation space at the current cursor location.

Remark

Added in version 3.92

Syntax

EditPaste

GetRow

Description

Returns an Integer representing the row position of the cursor.

Syntax

GetRow()

GetCol

Description

Returns an Integer representing the column position of the cursor.

Syntax

GetCol()

GetColor

Description

Returns an Integer representing the color code at the given screen location.

Syntax

GetColor(row, col)

The **GetColor** function syntax has these parts:

Part	Type	Description
row	Integer	The row where the color attribute is read.
col	Integer	The column where color attribute is read.

GetColor Return Values.

The following values are returned:

0 – default Color/Green

1 – Blue

2 – Red

3 – Pink

4 – Green

5 – Turquoise

6 – Yellow

7 – White

GetIpAddress

Description

This function returns the IP address of the computer running Quick3270.

Syntax

String = **GetIpAddress**(index)

The **GetIpAddress** function syntax has these parts:

Part	Type	Description
index	Integer	Required. As a Pc can have several network adapters, it can have several IP addresses. So the GetIpAddress function uses an index, the first address is on index = 0. It's possible to get all the IP addresses by incrementing the index until a zero length string is returned. This version returns only IPv4 addresses.

Remarks

Added in version 4.21.

GetMacroKeystroke

Description

This function returns the Virtual Keycode of the keystroke that started the macro. It returns 0 in case the macro was started from the menu.

Syntax

KeyCode = **GetMacroKeystroke**()

The list of the Keycodes:

[http://msdn.microsoft.com/de-de/library/0z084th3\(v=vs.90\).aspx](http://msdn.microsoft.com/de-de/library/0z084th3(v=vs.90).aspx)

The Upper word contains the modifier (Alt / Ctrl / Shift)

Sample:

```
WshShell = CreateObject("WScript.Shell")
Result = GetMacroKeystroke()

KeyStr$ = "Key pressed: "

if Result And &H100 Then
    KeyStr$ = KeyStr$ & "Alt + "
endif

if Result And &H200 Then
    KeyStr$ = KeyStr$ & "Ctrl + "
endif

if Result And &H400 Then
    KeyStr$ = KeyStr$ & "Shit + "
endif

' Remove the modifier value from Result and check the VK_KEY value
Result = Result And &HFF

if Result = 0 Then
    KeyStr$ = KeyStr$ & "None"
endif

if Result = &H70 Then
    KeyStr$ = KeyStr$ & "F1"
endif

if Result = &H71 Then
    KeyStr$ = KeyStr$ & "F2"
endif

WshShell.popup(KeyStr$)

End
```

Remarks

Added in version 4.30.

GetOIA

Description

Returns the string representing the content of the OIA line.

Syntax

String **GetOIA**()

GetString

Description

Returns the text from the specified screen location.

Syntax

GetString(*row, col, length*)

The **GetString** function syntax has these arguments:

Part	Type	Description
row	Integer	The row where the text string begins.
col	Integer	The column where the text string begins.
length	Integer	The length of the text string..

Remarks

Null and non-printable characters are replaced by spaces.

Added in version 4.12

GetTextRect

Description

Returns characters from a rectangular area in the presentation space. No wrapping takes place in the text retrieval; only the rectangular area is retrieved.

Syntax

GetTextRectfunction (*StartRow, StartCol, EndRow, EndCol*)

The **GetTextRect** function syntax has these arguments:

Part	Type	Description
StartRow	Integer	Row at which to begin the retrieval in the presentation space.
StartCol	Integer	Column at which to begin the retrieval in the presentation space.
EndRow	Integer	Row at which to end the retrieval in the presentation space
EndCol	Integer	Column at which to end the retrieval in the presentation space..

Remarks

Null and non-printable characters are replaced by spaces.
Added in version 4.31

GetTimeoutValue

Description

Returns the number of milliseconds used by Wait operations.

Syntax

GetTimeoutValue()

GetVisible

Description

Returns the visible state of the terminal Window.

Syntax

GetVisible()

Returns

The function returns **True** if application Window is Visible.

GetWindowTitle

Description

Returns the title of the terminal Window.

Syntax

GetWindowTitle()

MoveTo

Description

Moves the cursor to the specified screen location.

Syntax

MoveTo(*row*, *col*)

The **MoveTo** function syntax has these arguments:

Part	Type	Description
row	Integer	The row location.
col	Integer	The column location.

MoveRelative

Description

Moves the cursor relative from the current screen location.

Syntax

MoveRelative(*row*, *col*)

The **MoveRelative** function syntax has these arguments:

Part	Type	Description
row	Integer	The number of rows to move.
col	Integer	The number of columns to move.

OpenPrinter

Description

Allocates the default Windows printer for the PrintScreen function.

Syntax

OpenPrinter

Remark

OpenPrinter must be called before the PrintScreen function.

ClosePrinter

Description

Frees the resources allocated by the OpenPrinter function.

Syntax

ClosePrinter

PrintScreenFontSize

Description

PrintScreenFontSize is a global variable used to specify the font size used by the PrintScreen function.

If this variable is not set or set to 0, the default font size is used.

Syntax

PrintScreenFontSize = 8.

PrintScreen

Description

Prints the presentation space.

Syntax

PrintScreen [*Options*]

The options must be separated by a space character.

Valid options are:

BW: Print in Black & White

HEADER: Print the page header (overwrites the default settings)

NOHEADER: Don't print the page header (overwrites the default settings)

FORMFEED: Execute a form feed before the print job

Sample

```
OpenPrinter
PrintScreen
PrintScreen BW
PrinterFontSize=8
PrintScreen NOHEADER FORMFEED
PrinterFontSize=0
PrintScreen
ClosePrinter
End
```

PutString

Description

Puts text in the specified location on the screen.

Syntax

PutString *str* [,*row* , *col*]

The **PutString** syntax has these arguments:

Part	Type	Description
str	String	Required. Text that you want to put on the screen.
row	Integer	Optional. Integer specifying the row in which to put the text.
col	Integer	Optional. Integer specifying the column in which to put the tex.

Remark

If row and column are omitted, the current caret location is used.

ReceiveFile

Description

This function allows to receive a file from a host session.

Syntax

ReceiveFile *PcFile HostFile Options*

or

Boolean **ReceivedFile**(*PcFile,HostFile,Options*)

The **ReceiveFile** syntax has these arguments:

Part	Type	Description
PcFile	String	Required. String expression, pc file name.
HostFile	String	Required. String expression, host file name
Options	String	Optional. File transfer options

Remarks

Function added in version 3.98

Following options are supported

- ASCII
- CRLF
- APPEND
- LRECL
- RECFM
- CLEAR/NOCLEAR
- PROGRESS
- QUIET
- VM/CMS/MVS/TSO/CICS

Other supported options are the same as the EHLAPI Receive File function.

Search

Description

Search for the first occurrence of text in the Presentation Space.

Syntax

Boolean **Search**(*Token* [,*row*, *col*])

The **Search** syntax has these arguments:

Part	Type	Description
Token	String	Required. String to search for.
row	Integer	Optional. Row position to start search in Presentation Space
col	Integer	Optional. Column position to start search in Presentation Space

Remark

Function added in version 4.12

SendDarkKeys

Description

Puts text in the specified location on the screen. The text is encrypted.

Syntax

SendDarkKey str

The **SendDarkKey** syntax has these arguments:

Part	Type	Description
str	String	Required. String expression that you want to put on the screen.

Remarks

The **SendDarkKey** is used only by the macro recorder. All text typed in a "hidden" field will be stored encrypted. The SendDarkKey allows reading this encrypted text and sending the original text back to the emulator. This function is not intended to be used by the user. However you can Copy / Paste a **SendDarkKey** statement generated by the macro recorder into another macro file

SendFile

Description

This function allows to send a file to a host session.

Syntax

SendFile *PcFile HostFile Options*

or

Boolean **SendFile**(*PcFile,HostFile,Options*)

The **SendFile** syntax has these arguments:

Part	Type	Description
PcFile	String	Required. String expression, pc file name.
HostFile	String	Required. String expression, host file name
Options	String	Optional. File transfer options

Remarks

Function added in version 3.98

Following options are supported

- ASCII
- CRLF
- APPEND
- LRECL
- RECFM
- CLEAR/NOCLEAR
- PROGRESS
- QUIET
- VM/CMS/MVS/TSO/CICS

Other supported options are the same as the EHLLAPI Send File function.

SendKeys

Description

Send keystrokes at the current cursor location.

Syntax

SendKeys *str*

The **SendKeys** syntax has these arguments:

Part	Type	Description
str	String	Required. String expression that you want to put on the screen.

Remarks

Host function keys can be specified with mnemonics.

- Mnemonics must be enclosed in angle.
- Mnemonics are case insensitive

Table of mnemonics representing applicable function keys.

Description	Mnemonics	Emulation
Alternate Cursor	<AltCursor>	3270 and 5250
Attention	<Attn>	3270 and 5250
Back Space	<Backspace>	3270 and 5250
Back Tab	<Backtab>	3270 and 5250
Back Word Tab	<BackWord>	3270 and 5250
CapsLock	<CapsLock>	3270 and 5250
Clear	<Clear>	3270 and 5250
Cursor down	<Down>	3270 and 5250
Cursor left	<Left>	3270 and 5250
Cursor right	<Right>	3270 and 5250
Cursor up	<Up>	3270 and 5250
Cursor left fast	<Left2>	3270 and 5250
Cursor right fast	<Right2>	3270 and 5250
Cursor Select	<CursorSelect>	3270 and 5250
Delete char	<Delete>	3270 and 5250
Duplicate	<Dup>	3270 and 5250
Edit-Select All	<Edit-Select All>	3270 and 5250

Description	Mnemonics	Emulation
Edit-Cut	<Edit-Cut>	3270 and 5250
Edit-Copy	<Edit-Copy>	3270 and 5250
Edit-Paste	<Edit-Paste>	3270 and 5250
Edit-Paste Continue	<Edit-Paste Continue>	3270 and 5250
Edit-Copy Append	<Edit-Copy Append>	3270 and 5250
Edit-Undo Paste	<Edit-Undo Paste>	3270 and 5250
Enter	<Enter>	3270 and 5250
Erase EOF	<EraseEOF>	3270 and 5250
Erase Input	<EraseInput>	3270 and 5250
Erase Field	<EraseField>	3270 and 5250
End of Field	<EOF>	3270 and 5250
Field Exit	<Field Exit>	5250
Field Minus	<Field ->	5250
Field Plus	<Field +>	5250
Field Mark	<FieldMark>	3270 and 5250
Forward Word Tab	<ForwardWord>	3270 and 5250
Help	<Help>	5250
Home	<Home>	3270 and 5250
Host Print	<Host Print>	5250
Insert Toggle	<Insert>	3270 and 5250
Jump to next session	<Jump>	3270 and 5250
Last Field	<Last Field>	3270 and 5250
New Line	<NewLine>	3270 and 5250
Num Lock	<NumLock>	3270 and 5250
Page Down or Roll Up	<Page Down>	5250
Page Up or Roll Down	<Page Up>	5250
Print Screen - Default	<PrintScreen>	3270 and 5250
Print Screen - File	<PrintScreen to File>	3270 and 5250
Print Screen - Printer	<PrintScreen to Printer>	3270 and 5250
Reset	<Reset>	3270 and 5250
Ruler Toggler	<Rule>	3270 and 5250

Description	Mnemonics	Emulation
Scroll Lock	<ScrLock>	3270 and 5250
System Request	<SysReq>	3270 and 5250
Tab	<Tab>	3270 and 5250
Pa1	<Pa1>	3270 and 5250
Pa2	<Pa2>	3270 and 5250
Pa3	<Pa3>	3270 and 5250
Pf1 to Pf24	<Pf1> – Pf24>	3270 and 5250

Sample

SendKeys "Logon<Enter>"

SetText

Description

Send the given text to the current field.
Unlike SendKeys, mnemonics are not supported.

Syntax

SetText *str*

The **SetText** syntax has this argument:

Part	Type	Description
str	String	Required. String expression to set in field.

Remarks

Function added in version 4.30

SetTimeoutValue

Description

Specifies the number of milliseconds used by Wait operations.

Syntax

SetTimeoutValue *delay*

The **SetTimeoutValue** syntax has these arguments:

Part	Type	Description
delay	Integer	Required. the number of milliseconds.

SetVisible

Description

Sets the visible state of the terminal Window.

Syntax

SetVisible *state*

The **SetVisible** syntax has these arguments:

Part	Type	Description
state	Boolean	Required. TRUE if visible, FALSE if invisible.

SetWindowTitle

Description

Sets the application title bar text.

Syntax

SetWindowTitle *title*

The **SetWindowTitle** syntax has these arguments:

Part	Type	Description
title	String	Required. String to be displayed in the title bar.

Updated

Description

Returns **True** if the screen was updated since the last call of this function.

Syntax

Updated()

WaitForAttrib

Description

Waits until the specified attribute is displayed at the specified screen location. Returns True if the Attribute Value is found,

Syntax

Boolean **WaitForAttrib**(row, col, Waitdata, [optional] MaskData, [optional] plane , [optional] TimeOut, [optional] WaitForIr)

The **WaitForAttrib** function syntax has these arguments:

Part	Type	Description
row	Integer	Required. Row position of the attribute.
col	Integer	Required. Column position of the attribute.
Waitdata	Integer	Required. Specifies the value of the attribute to wait for.
MaskData	Integer	Optional. Specifies the value to use as a mask with the attribute. The default value is &HFF.
plane	Integer	Optional. Specifies the plane of the attribute to get. The plane can have the following values: 1 Text Plane 2 Color Plane 3 Field Plane (default) 4 Extended Field Plane The default value is 3.
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue
WaitForIr	Boolean	Optional. If this value is true, after meeting the wait condition the function will wait until the session is ready to accept input. Default value is False

Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

WaitForCursor

Description

Waits until the cursor is at the specified screen location

Syntax

WaitForCursor(*row, col, [optional] TimeOut*)

The **WaitForCursor** function has these arguments:

Part	Type	Description
row	Integer	Required. Row position of the cursor.
col	Integer	Required. Column position of the cursor.
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue.

Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

WaitForCursorMove

Description

Waits until the cursor has moved the specified number of rows and columns from its current position.

Syntax

Boolean **WaitForCursorMove**(*row, col, [optional] TimeOut*)

The **WaitForCursorMove** function syntax has these arguments:

Part	Type	Description
row	Integer	Required. The number of rows to move..
col	Integer	Required. The number of columns to move..
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue.

Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

WaitForKbdUnlock

Description

The function waits until the OIA of the connection indicates that the connection is able to accept keyboard input.

Syntax

Boolean **WaitForKbdUnlock**(*[optional] TimeOut*)

The **WaitForKbdUnlock** function syntax has this argument:

Part	Type	Description
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue.

Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

WaitForString

Description

Waits until the specified text appears on the specified screen location. Search string is case-sensitive.

Syntax

Boolean **WaitForString**(*str, row, col, [optional] TimeOut*)

The **WaitForString** function syntax has these arguments:

Part	Type	Description
str	String	Required. The text string that you want to wait for.
row	Integer	Required. The row where you expect the string to appear.
col	Integer	Required. The column where you expect the string to appear.
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue.

Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

WaitHostQuiet

Description

Waits for the host to not send data for a specified number of milliseconds.

Syntax

Boolean **WaitHostQuiet**(*settletime*)

The **WaitHostQuiet** function syntax has these arguments:

Part	Type	Description
settletime	Integer	Required. The amount of time, in milliseconds, that the host should remain "quiet."

Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

Functions supported in IBM® Personal Communications™ compatibility mode

The possibility to run IBM® Personal Communications™ macros has been added in version **3.95** of Quick3270.

The IBM® Pers. Comm. compatibility mode allows Quick3270 to run VBScript and macros recorded by Pers. Comm.

The following conditions must be verified to recognize an IBM® Pers. Comm. VBScript:

- File extension must be .mac
- VBScript Macros must start with the following header:

```
[PCOMM SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=xxx
[PCOMM SCRIPT SOURCE]
OPTION EXPLICIT
```

Quick3270 can only run macros that use functions used by the IBM® Pers. Comm. macro recorder.

The standard PComm™ COM objects must not be created by the macro. They are available by default (autECLSession, autECLOIA, autECLPS,...).

autECLSession.SetConnectionByName

Description

This function is simply ignored by Quick3270. With Quick3270 it's not possible to connect to another session.

Syntax

autECLSession.SetConnectionByName(name)

The **autECLSession.SetConnectionByName** syntax has this argument:

Part	Type	Description
name	String	Required. One-character string short name of the connection (A-Z).

autECLSession.autECLOIA.WaitForAppAvailable

Description

For compatibility only, this function is simply ignored by Quick3270.

Syntax

WaitForAppAvailable (*[optional] TimeOut*)

Part	Type	Description
TimeOut	Integer	Optional. The maximum length of time in milliseconds to wait, The default is Infinite.

Sample

```
bResult = autECLSession.autECLOIA.WaitForAppAvailable(1000)
```

autECLSession.autECLOIA.WaitForInputReady

Description

The WaitForInputReady method waits the connection is able to accept keyboard input. This function is similar to the Quick3270 WaitForKbdUnlock function

Syntax

WaitForInputReady (*[optional] TimeOut*)

Part	Type	Description
TimeOut	Integer	Optional. The maximum length of time in milliseconds to wait, The default is Infinite.

Sample

```
bResult = autECLSession.autECLOIA.WaitForInputReady(1000)
```

autECLSession.autECLXfer.ReceiveFile

Description

This function allows receiving a file from a host session.

Syntax

autECLSession.autECLXfer.ReceiveFile *PcFile HostFile Options*

The **ReceiveFile** syntax has these arguments:

Part	Type	Description
PcFile	String	Required. Pc file name.
HostFile	String	Required. Host file name
Options	String	Optional. File transfer options

Remarks

Property added in version 3.98

Following options are supported

- ASCII
- CRLF
- APPEND
- LRECL
- RECFM
- CLEAR/NOCLEAR
- PROGRESS
- QUIET
- VM/CMS/MVS/TSO/CICS

Other supported options are the same as the options in EHLLAPI Receive File function.

autECLSession.autECLXfer.SendFile

Description

This function allows sending a file to a host session.

Syntax

autECLSession.autECLXfer.SendFile *PcFile HostFile Options*

The **SendFile** syntax has these arguments:

Part	Type	Description
PcFile	String	Required. Pc file name.
HostFile	String	Required. Host file name
Options	String	Optional. File transfer options

Remarks

Property added in version 3.98

Following options are supported

- ASCII
- CRLF
- APPEND
- LRECL
- RECFM
- CLEAR/NOCLEAR
- PROGRESS
- QUIET
- VM/CMS/MVS/TSO/CICS

Other supported options are the same as the options in EHLLAPI Send File function.

autECLSession.autECLPS.CursorPosRow

Description

Returns current row position of the cursor in the presentation space.

Syntax

CurPosRow = autECLSession.autECLPS.CursorPosRow

Remarks

Property added in version 4.31

autECLSession.autECLPS.CursorPosCol

Description

Returns current column position of the cursor in the presentation space.

Syntax

CurPosCol = autECLSession.autECLPS.CursorPosCol

Remarks

Property added in version 4.31

autECLSession.autECLPS.GetTextRect

Description

Returns characters from a rectangular area in the presentation space.
No wrapping takes place in the text retrieval; only the rectangular area is retrieved.

Syntax

autECLSession.autECLPS.GetTextRect (*StartRow, StartCol, EndRow, EndCol*)

The **GetTextRect** method has these arguments:

Part	Type	Description
StartRow	Integer	Row at which to begin the retrieval in the presentation space.
StartCol	Integer	Column at which to begin the retrieval in the presentation space.
EndRow	Integer	Row at which to end the retrieval in the presentation space
EndCol	Integer	Column at which to end the retrieval in the presentation space..

Remarks

Null and non-printable characters are replaced by spaces.

Function added in version 4.31

autECLSession.autECLPS.SetText

Description

Send the given text to the current field.
Unlike SendKeys, mnemonics are not supported.

Syntax

autECLSession.autECLPS.SetText *str*

The **autECLSession.autECLPS.SetText** syntax has this argument:

Part	Type	Description
str	String	Required. Text to set in field.

Remarks

Function added in version 4.30

autECLSession.autECLPS.SendKeys

Description

Send keystrokes at the current cursor location.

Syntax

autECLSession.autECLPS.SendKeys *str*

The **autECLSession.autECLPS.SendKeys** method has these arguments:

Part	Type	Description
str	String	Required. Text that you want to put on the screen.

Remarks

Host function keys can be specified with mnemonics.

- Mnemonics must be enclosed in square brackets.
- Mnemonics are case insensitive

Table of mnemonics representing applicable function keys.

Description	Mnemonics IBM PComm™ compatibility mode	Emulation
-------------	---	-----------

Description	Mnemonics IBM PComm™ compatibility mode	Emulation
Alternate Cursor	[altcsr]	3270 and 5250
Attention	[attn]	3270 and 5250
Back Space	[backspace]	3270 and 5250
Back Tab	[backtab]	3270 and 5250
Back Word Tab	[wordleft]	3270 and 5250
CapsLock	[capslock]	3270 and 5250
Clear	[clear]	3270 and 5250
Cursor down	[down]	3270 and 5250
Cursor left	[left]	3270 and 5250
Cursor right	[right]	3270 and 5250
Cursor up	[up]	3270 and 5250
Cursor left fast	[fastleft]	3270 and 5250
Cursor right fast	[fastright]	3270 and 5250
Cursor Select	[crsel]	3270 and 5250
Delete char	[delete]	3270 and 5250
Duplicate	[dup]	3270 and 5250
Edit-Select All		3270 and 5250
Edit-Cut		3270 and 5250
Edit-Copy		3270 and 5250
Edit-Paste		3270 and 5250
Edit-Paste Continue		3270 and 5250
Edit-Copy Append		3270 and 5250
Edit-Undo Paste		3270 and 5250
Enter	[enter]	3270 and 5250
Erase EOF	[eraseeof]	3270 and 5250
Erase Input	[erinp]	3270 and 5250
Erase Field		3270 and 5250
End of Field	[eof]	3270 and 5250
Field Exit	[fldext]	5250
Field Minus	[field+]	5250
Field Plus	[field-]	5250

Description	Mnemonics IBM PComm™ compatibility mode	Emulation
Field Mark	[fieldmark]	3270 and 5250
Forward Word Tab	[wordright]	3270 and 5250
Help	[help]	5250
Home	[home]	3270 and 5250
Host Print	[print]	5250
Insert Toggle	[insert]	3270 and 5250
Jump to next session	[jump]	3270 and 5250
Last Field		3270 and 5250
New Line	[newline]	3270 and 5250
Num Lock	[numlock]	3270 and 5250
Page Down or Roll Up	[pagedn]	5250
Page Up or Roll Down	[pageup]	5250
Print Screen - Default	[printps]	3270 and 5250
Print Screen - File		3270 and 5250
Print Screen - Printer		3270 and 5250
Reset	[reset]	3270 and 5250
Ruler Toggler		3270 and 5250
Scroll Lock	[scrlock]	3270 and 5250
System Request	[sysreq]	3270 and 5250
Tab	[tab]	3270 and 5250
Pa1	[pa1]	3270 and 5250
Pa2	[pa2]	3270 and 5250
Pa3	[pa3]	3270 and 5250
Pf1 to Pf24	[pf1] - [pf24]	3270 and 5250

Sample

```
autECLSession.autECLPS.SendKeys "[enter]"
```

autECLSession.autECLPS.Wait

Description

The Wait method waits for the number of milliseconds specified by the milliseconds parameter.

Syntax

autECLSession.autECLPS.Wait (*Delay*)

The **autECLSession.autECLPS.Wait** method has this argument:

Part	Type	Description
Delay	Integer	The number of milliseconds to wait.

Sample

```
autECLSession.autECLPS.Wait 4860
```

autECLSession.autECLPS.WaitForAttrib

Description

Waits until the specified attribute is displayed at the specified screen location. Returns True if the Attribute value is found,

Syntax

autECLSession.autECLPS.WaitForAttrib(row, col, Waitdata, [optional] MaskData, [optional] plane , [optional] TimeOut, [optional] WaitForIr)

The **autECLSession.autECLPS.WaitForAttrib** method has these arguments:

Part	Type	Description
row	Integer	Required. Row position of the attribute.
col	Integer	Required. Column position of the attribute.
Waitdata	Integer	Required. Specifies the value of the attribute to wait for.
MaskData	Integer	Optional. Specifies the value to use as a mask with the attribute. The default value is &HFF.
plane	Integer	Optional. Specifies the plane of the attribute to get. The default value is 3.
TimeOut	Integer	Optional. Specifies the maximum length of time in Milliseconds to wait. Default value is Infinite
WaitForIr	Boolean	Optional. If this value is true, after meeting the wait condition the function will wait until the session is ready to accept input. Default value is False

Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

Remarks

The plane can have the following values:

- 1 Text Plane
- 2 Color Plane
- 3 Field Plane
- 4 Extended Field Plane

Sample

```
autECLSession.autECLPS.WaitForAttrib 19,28,"00","3c",3,1000
```

autECLSession.autECLPS.WaitForCursor

Description

Waits until the cursor is at the specified screen location.

Syntax

autECLSession.autECLPS.WaitForCursor(*row, col, [optional TimeOut]*)

The **autECLSession.autECLPS.WaitForCursor** method has these arguments:

Part	Type	Description
row	Integer	Required. Row position of the cursor.
col	Integer	Required. Column position of the cursor.
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is Infinite

Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

autECLSession.autECLPS.WaitForString

Description

Waits until the specified text appears on the specified screen location. Search string is case-sensitive.

Syntax

autECLSession.autECLPS.WaitForString(*str, row, col, [optional] TimeOut, [optional] WaitForIr, [optional] bCaseSens*)

The **WaitForString** function syntax has these arguments:

Part	Type	Description
str	String	Required. The text string that you want to wait for.
row	Integer	Required. The row where you expect the string to appear. Unlike PComm this parameter is not optional and value 0 for entire screen searches is not supported
col	Integer	Required. The column where you expect the string to appear. Unlike PComm this parameter is not optional and value 0 for entire screen searches is not supported.
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue.

WaitForIr	Boolean	Optional. If this value is true, after meeting the wait condition the function will wait until the session is ready to accept input. Default value is False This parameter is ignored by Quick3270
bCaseSens	Boolean	Optional. If this value is True, the wait condition is verified as case-sensitive. Default value is False This parameter is ignored by Quick3270

Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

autSystem.Inputnd

Description

Displays a popup input box to the user with a no-display text box so that when the user types in data only asterisks(*) are displayed.

Syntax

autSystem.Inputnd()

Returns

The characters typed into the input box.

Remarks

Function added in version 4.30

Functions supported in Attachmate's Extra!™ compatibility mode

The possibility to run Attachmate's Extra!™ macros has been added in version **3.96** of Quick3270.

The Attachmate's Extra!™ compatibility mode allows Quick3270 to run directly macros recorded by Extra!™.

The following conditions must be verified to recognize an Extra!™ macro:

- File extension must be .ebm or .txt
- For .ebm files, the macro must have the standard Extra! macro header. Quick3270 will make some pointer and keyword check to identify an Extra! macro header
- For .txt files, no checks are made. Quick3270 assumes that all .txt files are Attachmate's Extra!™ macros

Compatibility restrictions:

- Quick3270 can only run macros that uses functions used by the Extra!™ macro recorder.
- System and Sess0 objects are default names created by the Extra! Macro recorder. Quick3270 uses the same name (not modifiable) for compatibility. The user should not change the default name of the objects or create Extra! Objects with other names: This is not supported by Quick3270.
- SendFile and ReceiveFile statements are supported, however Extra! file transfer schemes are not supported.

List of supported methods

- Sess0.Screen.GetString
- Sess0.Screen.MoveTo
- Sess0.Screen.PutString
- Sess0.Screen.Rows
- Sess0.Screen.Sendkeys
- Sess0.Screen.Updated
- Sess0.Screen.WaitForCursor
- Sess0.Screen.WaitForCursorMove
- Sess0.Screen.WaitForHostQuiet
- Sess0.Screen.WaitForString
- Sess0.TimeOutValue
- Sess0.Visible
- Sess0.FileTransferHostOS
- Sess0.SendFile
- Sess0.ReceiveFile

List of ignored lines (the objects are created by default)

- Dim Sessions As Object
- Dim Sess0 As Object
- Dim System As Object
- Set System = CreateObject("EXTRA.System")
- Set Sessions = System.Sessions
- Set Sess0 = System.ActiveSession
- Sess0.FileTransferScheme

Macro Samples

Select an Excel file using GetOpenFileName

```
;Create objects
WshShell = CreateObject("WScript.Shell")

strFileName = GetOpenFileName("Select File", "", "Excel Files *.xls|*.xls")
WshShell.popup(strFileName)
WshShell = Nothing
End
```

Creating a Reference to an Excel Object

```
;Create objects
appExcel = CreateObject("Excel.Application")
appExcel.Visible = True
Set iLine = 1
appExcel.activesheet.cells(2,1).value = GetString(iLine+6, 6, 8)
...
appExcel = Nothing
End
```

File I/O

```
iFile = Freefile
sFile = "C:dummy.txt"
sTmp = "      "
result = MsgBox ("sFile=" & sFile , vbOK , "Continue")
result = MsgBox ("iFile=" & str(iFile) , vbOK , "Continue")
Open sFile For Input as iFile
result = MsgBox ("About to prime read",vbOK,"Continue")
LineInput #iFile, sTmp
result = MsgBox ("First read done",vbOK,"Continue")
While NOT EOF (iFile)
    MsgBox sTmp, vbOK , "Processing"
    LineInput #iFile, sTmp
Wend
Close #iFile
result = MsgBox ("Finished",vbOK,"Continue")
End
```

Connecting to a data source and executing SQL Queries

```
connectSQL = CreateObject("ADODB.Connection")
recordsetSQL = CreateObject("ADODB.RecordSet")
recordsetSQL.ActiveConnection = connectSQL
recordsetSQL.Source = "Select * from myRange1"
recordsetSQL.Open
Adrs_L = recordsetSQL.Fields.Item("A1").value
...
End
```

Find Values from screen and use the WScript object

```
;Create objects and init variables
WshShell = CreateObject("WScript.Shell")
Screen=""
sFind=0
FIN=""
;Find Values from screen
Screen = GetString(1,1,1920)
sFind = inStr(Screen,"LSFYFND")
FIN=MID(Screen,(sFind + 9),5)
WshShell.popup(sFind)
if sFind > 0 then
    sFind = inStr(Screen,"IVGFINAL")
    FIN = MID(Screen,(sFind + 9),5)
endif

;Show Popup Value
WshShell.popup(FIN)

;Destroy Open Objects and init Variables
WshShell = Nothing
Screen=""
FIN=""
sFind=0
End
```

Simple logon script

```
HostSettleTime = 3000 ; milliseconds
Result = WaitForCursor(19, 64)
SendKeys "logon<Enter>"
Result = WaitForCursor(5, 40)
SetWindowTitle "Macro" + " Quick3270"
sPassword = PasswordBox("Enter Password", "Logon" )
SendKeys "USERNAME"
b = MoveTo(6,40)
SendKeys sPassword
SendKeys "<Enter>"
End
```

Using LDAP to retrieve the user name and password and start logon script

```
Set QuickExit = False
Set jRoot = GetObject("LDAP://RootDSE")
DomainPath = jRoot.Get("DefaultNamingContext")
Set Domain = GetObject("LDAP://" + DomainPath)
DomainPath = "LDAP://s1-mes.intra.company.com/DC=intra,DC=company,DC=com"
Set con = CreateObject("ADODB.Connection")
Set Com = CreateObject("ADODB.Command")
con.Provider = "ADsDSOObject"
con.Open "Active Directory Provider"
Set Com.ActiveConnection = con
Com.CommandText = "select HostUser, HostPassword from '" + DomainPath + "'
WHERE objectClass='user' AND sAMAccountName='" + sAMAccountName + "'"
Set rs = Com.Execute
if Not rs.EOF then
    HostUserName = rs.Fields("HostUserName")
    HostPassword = rs.Fields("HostPassword")
    if Len(HostUserName) = 0 Then
        rc = msgbox("User name unknown" , vbokonly + vbExclamation, "Error")
    else
        ; We have the user name and password. Start the logon process
        Result = WaitForString("Welcome", 1, 38)
        if Result = True then
            Result = MoveTo(29,47)
            SendKeys HostUserName.Value
            Result = MoveTo(30,47)
            SendKeys HostPassword.Value
            SendKeys "<Enter>"
            Result = WaitForKbdUnlock()
            r = GetString(28, 2, 8)
            if r = "rejected" then
                rc = msgbox("User already logged in" , vbokonly +
vbExclamation, "Error")
                Set QuickExit = True
            endif
        endif
    endif
endif

con.Close
rs = Nothing
Com = Nothing
con = Nothing
Domain = Nothing
jRoot = Nothing
End
```

PComm VBScript sample supported by Quick3270

```
[PCOMM SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=VBScript
[PCOMM SCRIPT SOURCE]
OPTION EXPLICIT
autECLSession.SetConnectionByName(ThisSessionName)

REM This line calls the macro subroutine
subSub1_

sub subSub1_()
    autECLSession.autECLIOIA.WaitForAppAvailable

    autECLSession.autECLIOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "a"
    autECLSession.autECLIOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "[enter]"

    autECLSession.autECLPS.WaitForAttrib 24,5,"00","3c",3,10000

    autECLSession.autECLPS.Wait 2937

    autECLSession.autECLIOIA.WaitForAppAvailable

    autECLSession.autECLIOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "[pf1]"

    autECLSession.autECLPS.WaitForAttrib 24,80,"08","3c",3,10000

    autECLSession.autECLPS.WaitForCursor 1,1,10000

    autECLSession.autECLIOIA.WaitForAppAvailable

    autECLSession.autECLIOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "/rcl"
    autECLSession.autECLIOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "[enter]"

    autECLSession.autECLPS.WaitForAttrib 24,5,"00","3c",3,10000

    autECLSession.autECLPS.Wait 4860

    autECLSession.autECLIOIA.WaitForAppAvailable

    autECLSession.autECLIOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "[pf3]"
end sub
```

PComm Macro sample supported by Quick3270

```
Description =Test Macro
run /min notepad.exe
[wait app]
[wait inp inh]
"a
[enter]
[wait inp inh]
wait 10 sec until FieldAttribute 0000 at (24,5)
wait 10 sec until cursor at (24,6)
[wait app]
[wait inp inh]
"sos
[enter]
[wait inp inh]
wait 10 sec until FieldAttribute 0008 at (24,80)
wait 10 sec until cursor at (1,1)
[wait app]
[wait inp inh]
"logon
[enter]b
```

Attachmate's Extra! macro sample supported by Quick3270

```
'-----
' This macro was created by the Macro Recorder.
' Session Document: "DISPLAY LOCALHOST.EDP"
' Date: Thursday, November 08, 2009 15:19:52
' User: Denis
'-----

' Global variable declarations
Global g_HostSettleTime%

Sub Main()
'-----
' Get the main system object
  Dim Sessions As Object
  Dim System As Object
  Set System = CreateObject("EXTRA.System") ' Gets the system object
  If (System is Nothing) Then
    MsgBox "Could not create the EXTRA System object."
    STOP
  End If
  Set Sessions = System.Sessions

  If (Sessions is Nothing) Then
    MsgBox "Could not create the Sessions collection object."
    STOP
  End If
'-----

' Set the default wait timeout value
  g_HostSettleTime = 3000      ' milliseconds

  OldSystemTimeout& = System.TimeoutValue
  If (g_HostSettleTime > OldSystemTimeout) Then
    System.TimeoutValue = g_HostSettleTime
  End If

' Get the necessary Session Object
  Dim Sess0 As Object
  Set Sess0 = System.ActiveSession
  If (Sess0 is Nothing) Then
    MsgBox "Could not create the Session object.  Stopping macro
    playback."
    STOP
  End If
  If Not Sess0.Visible Then Sess0.Visible = TRUE
  Sess0.Screen.WaitHostQuiet(g_HostSettleTime)

' This section of code contains the recorded events
  Sess0.Screen.Sendkeys("F<Enter>")
  Sess0.Screen.WaitHostQuiet(g_HostSettleTime)
  Sess0.Screen.Sendkeys("<Pf3>")
  Sess0.Screen.WaitHostQuiet(g_HostSettleTime)
  System.TimeoutValue = OldSystemTimeout
End Sub
```