PASSPORT
Technical Reference

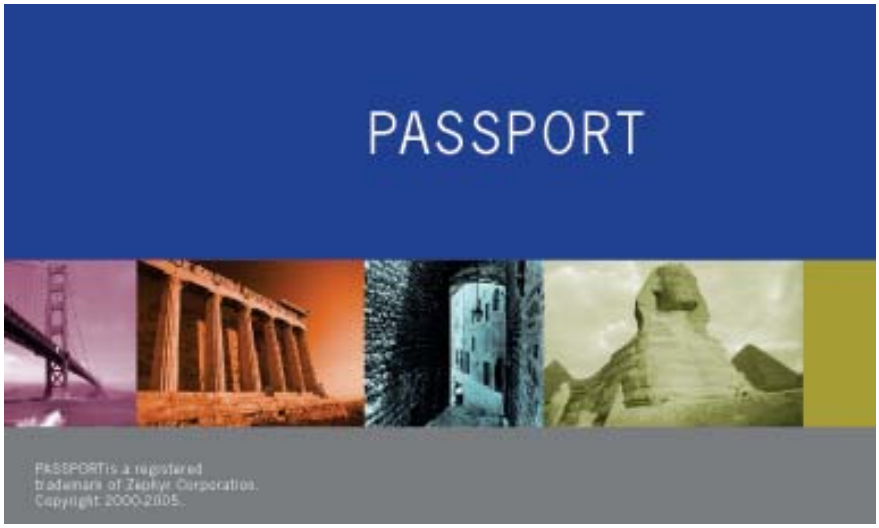

Table Of Contents

Introduction

Contacting Zephyr

Corporate Office
Zephyr Development Corporation
8 East Greenway Plaza, Suite 1414
Houston, Texas 77046-0801 USA

World Wide Web
http://www.zephyrcorp.com

Sales Information
sales@zephyrcorp.com

Technical Support
http://www.zephyrcorp.com/support.htm

Telephone
(713) 623-0089
(800) 966-3270 Toll Free (United States and Canada)

Fax
(713) 623-0091

Zephyr EMEAI - Distributor for Europe, Middle East, Asia & India
71 High Street
Harrold
Bedfordshire
MK43 7BJ
United Kingdom

Sales Information
sales@integranet.co.uk

Technical Support
support@integranet.co.uk

Telephone
+44 (0) 1234 721755

Fax
+44 (0) 1234 721672

File Transfer

File Transfer Overview

File Transfer Method
You may choose either IND$FILE or FTP as the default **File Transfer Method** for a session. IND$FILE file transfer is available only for IBM mainframes using TN3270 or TN3270E and FTP may be used for any emulation type. For PASSPORT WEB TO HOST, the **File Transfer Method** may be selected in the WEB TO HOST Administrator by creating an attribute profile and choosing either IND$FILE or FTP on the Transfer tab. Then assign this attribute profile to the session profile. For PASSPORT WEB TO HOST and PC TO HOST clients, this is done using the **Options→Transfer** menu command. When FTP is selected, you must also configure the FTP Server Settings. Click the **FTP Server Settings** button and enter the appropriate Host Name, FTP Port, etc.

File Transfer Schemes
Schemes are used to configure the file transfer options for the session. The following default file transfer schemes are available, depending on the **File Transfer Method** selected:

IND$FILE:

- CMS Text Default
- CMS Binary Default
- TSO Text Default
- TSO Binary Default
- CICS Text Default
- CICS Binary Default
- Custom

FTP:

- FTP Default
- Custom

IND$FILE File Transfer
With IND$FILE selected as the **File Transfer Method**, the **Transfer→Send** and **Transfer→Receive** commands are used to transfer files between the host and PC. This will display the **IND$FILE file transfer Send/Receive** dialog box. Provide the **PC File Name** and **Host File Name**, select a **Transfer Scheme**, then click the Send/Receive button.

FTP File Transfer
With FTP selected as the **File Transfer Method**, the **Transfer→Send** and **Transfer→Receive** commands are used to transfer files between the host and PC. This will display the **FTP file transfer Send/Receive** dialog box. Provide the **PC File Name** and **Host File Name**, select a **Transfer Scheme**, then click the Send/Receive button.

PASSPORT FTP Client

The FTP file transfer client is a stand alone program that is launched from within PASSPORT from the **Transfer→FTP** menu command. FTP file transfer can be used with any host that has an FTP server running. For detailed information about the PASSPORT FTP Client, see the **Help→Help Topics** menu command in the FTP Client.

File Transfer Custom Parameters

This technical reference describes the free form text that may be entered into the custom file type parameters input field for file transfer options. The valid text that may be entered depends on the file transfer operation (either **Send** or **Receive**) and the mainframe operating system (either CICS, MVS/TSO or VM/CMS). These file transfer parameters are discussed in the following sections:

CICS Mainframe Operating System

- Send Parameters
- Receive Parameters

MVS/TSO Mainframe Operating System

- Send Parameters
- Receive Parameters

VM/CMS Mainframe Operating System

- Send Parameters
- Receive Parameters

File Transfer Host Messages

TRANS00 - Error in file transfer: file transfer canceled
An error occurred in the file transfer, which may be an error in the data being transferred, or an unidentified system error.

TRANS03 - File transfer complete
The file transfer operation has been successfully completed.

TRANS04 - File transfer complete with records segmented
The file transfer operation has been completed, and any record greater than the logical record length (LRECL) of the file being appended has been divided and becomes multiple records.

TRANS05 - Personal computer filespec incorrect: file transfer canceled
An error exists in the PC's file name.

TRANS06 - Command incomplete: file transfer canceled
You did not enter the required parameters after a SEND or RECEIVE command.

TRANS13 - Error writing file to host: file transfer canceled
The host program has detected an error in the file data during a RECEIVE operation.

TRANS14 - Error reading file from host: file transfer canceled
The host program has detected an error in the file data during a RECEIVE operation.

TRANS15 - Required host storage unavailable: file transfer canceled
You need 30 KB of main storage (not disk space) on the host for the file transfer, in addition to the host requirement.

TRANS16 - Incorrect request code: file transfer canceled
An invalid SEND or RECEIVE parameter was sent to the host.

TRANS17 - Missing or incorrect TSO data set name: file transfer canceled
You did not specify the TSO data set name, or the specified TSO data set name is not a sequential or partitioned data set.

TRANS17 - Invalid file name: file transfer canceled
The command that started the file transfer specified a file name that is not a valid file name for the host. Correct CICS file names must be 1 to 8 characters, composed of letters and numbers.

TRANS17 - Missing or incorrect CMS data set name: file transfer canceled
You did not specify the CMS data set name, or the specified CMS data set name is incorrect.

TRANS18 - Incorrect option specified: file transfer canceled
You specified an option that is invalid.

TRANS19 - Error while reading or writing to host disk: file transfer canceled
There is not enough space available for data on the host.

TRANS28 - Invalid option xxxxxxxx: file transfer canceled
You selected an option that is either not recognized, is specified as a positional keyword, or has an associated value that is incorrect.

TRANS29 - Invalid option xxxxxxxx with RECEIVE: file transfer canceled
You selected an option that is not valid with RECEIVE, but can be used with SEND.

TRANS30 - Invalid option xxxxxxxx with APPEND: file transfer canceled

You selected an option that is not valid with APPEND, but otherwise may be used.

TRANS31 - Invalid option xxxxxxxx without SPACE: file transfer canceled
You selected an option that can only be used if SPACE is also specified.

TRANS32 - Invalid option xxxxxxxx with PDS: file transfer canceled
You selected an option that is invalid with a host-partitioned data set.

TRANS33 - Only one of TRACKS, CYLINDERS, AVBLOCK allowed: file transfer canceled
SPACE can be specified in units of TRACKS, CYLINDERS, or AVBLOCK, and only one option can be used.

TRANS34 - CMS file not found: file transfer canceled
You did not specify an existing CMS file for RECEIVE.

TRANS35 - CMS disk is read-only: file transfer canceled
You specified a CMS file mode for the SEND key that does not allow write access.

TRANS36 - CMS disk is not accessed: file transfer canceled
You specified a CMS file mode that is not in the CMS search order.

TRANS37 - CMS disk is full: file transfer canceled
The CMS disk is full, or the maximum number of files on the minidisk (3400) has been reached, or the maximum number of data blocks per file (16060) has been reached.

TRANS99 - Host program error code xx xxxxxxxx: file transfer canceled
This is a host program error.

Mainframe System Requirements

Depending upon which IBM host system you use, one of the following IBM host-supported file transfer programs must be installed at your host site. The **Options** tab on the **File Transfer** dialog box is used to configure the mainframe operating system and the file transfer program name.

MVS/TSO Mainframe Operating System
IBM 3270-PC File Transfer Program Release 1 (5665-311)

At the TSO command level, type: IND$FILE. If the program is not installed, you will receive the message: [IKJ56500I] COMMAND IND$FILE NOT FOUND

VM/CMS Mainframe Operating System
IBM 3270-PC File Transfer Program Release 1 (5664-281)

The installation and support of the IBM host-supported file transfer programs is outside the scope of this documentation. A quick way to determine if the IBM file transfer product is installed on your host, is to do the following: In the CMS environment, type: IND$FILE. If the program is not installed, you will receive the message: UNKNOWN CP/CMS COMMAND

CICS Mainframe Operating System
Ask your systems administrator if the IBM IND$FILE host file transfer program is installed.

CICS

CICS Send Parameters

Send parameters are optional and are passed as parameters to the host IND$FILE program. The following parameters are valid for the CICS send file transfer operation:

ASCII
If the PC file consists of character data only, specify ASCII. This will cause the translation of all graphic and control characters from ASCII to EBCDIC during the transfer so that it is usable on the host computer.

BINARY
If the PC file consists of binary data, specify BINARY.

CRLF
Following the convention for ASCII text files, <CR> <LF> pairs are used to terminate records in the PC file, and a CTRL-Z (x'1A') marks the end of file. CRLF is a default option.

NOCRLF
Do not use <CR> <LF> pairs to terminate records in the CICS file.

CICS Receive Parameters

Receive parameters are optional and are passed as parameters to the host IND$FILE program. The following parameters are valid for the CICS receive file transfer operation:

APPEND
APPEND specifies that if the PC file exists, the CICS file contents are to be appended to it. Unless APPEND is specified, the PC file is replaced by the contents of the CICS file.

ASCII
If the CICS file consists of character data only, specify ASCII. This will cause the translation of all graphic and control characters from ASCII to EBCDIC during the transfer.

BINARY
If the PC file consists of binary data, specify BINARY.

CRLF
Following the convention for ASCII text files, <CR> <LF> pairs are used to terminate records in the PC file, and a CTRL-Z (x'1A') marks the end of file.

NOCRLF
Do not use <CR> <LF> pairs to terminate records in the PC file.

MVS/TSO

MVS/TSO Send Parameters

Send parameters are optional and are passed as parameters to the host IND$FILE program. The following parameters are valid for the MVS/TSO send file transfer operation:

ASCII
If the PC file consists of character data only, specify ASCII. This will cause the translation of all graphic and control characters from ASCII to EBCDIC during the transfer.

CRLF
Following the convention for ASCII text files, <CR> <LF> pairs are used to terminate records in the PC file, and a CTRL-Z (x'1A') marks the end of file. If the PC file consists of data other than text, such as binary format data, do not specify either ASCII or CRLF.

APPEND
If the data set already exists and is not empty, APPEND specifies that the contents of the PC file are to be appended to the data set. APPEND cannot be specified for members of a partitioned data set.

RECFM(x)
Specifies the record format of the data set. Replace x with either F, V, or U. F specifies that the data set contains fixed length records. V specifies that the data set contains variable length records. U specifies that the data set contains undefined length records. For existing data sets, the default value is the existing record format.

LRECL(n)
Specifies the logical record length (n) for a data set consisting of fixed length records or the maximum logical record length for a data set consisting of variable length records. Values permitted are 1 to 32760. For new data sets the default value is 80. This parameter is ignored if APPEND is specified. For data sets consisting of variable length records, the actual logical record length (i.e. the value stored in the Data Set Control Block) following the file transfer is set to the minimum of the value of LRECL and the longest record in the data set.

BLKSIZE(n)
Specifies the block size (n) for a new data set. For data sets containing fixed-length records, the block size must be a multiple of the record length. For data sets containing variable-length records, the block size must be greater than or equal to the record length plus four bytes. The block size must not exceed the track length of the device on which the data set resides. The block size must also not exceed 32,760 bytes. The default value is 3120. This parameter is ignored if the data set already exists.

SPACE(quantity[,increment])
Specifies the amount of disk space to be allocated for a new data set. If the SPACE parameter is used, you can use one of the three following options to specify the units used for quantity and increment:

- SAVBLOCK(value)
- TRACKS
- CYLINDERS

MVS/TSO Receive Parameters

Receive parameters are optional and are passed as parameters to the host IND$FILE program. When using the following positional parameters, care should be taken to include the "(" open parenthesis character and ")" close parentheses character in describing RECFM, LRECL, and BLKSIZE as denoted in the following example:

ASCII CRLF RECFM(F) LRECL(80) BLKSIZE(6160)

The following parameters are valid for the MVS/TSO receive file transfer operation:

ASCII
If the PC file consists of character data only, specify ASCII. This will cause the translation of all graphic and control characters from EBCDIC to ASCII during the transfer.

CRLF
Following the convention for ASCII text files, <CR> <LF> pairs are used to terminate records in the PC file, and a CTRL-Z (x'1A') marks the end of file. If the PC file consists of data other than text, such as binary format data, do not specify either ASCII or CRLF.

APPEND
If the data set already exists and is not empty, APPEND specifies that the contents of the Mainframe file are to be appended to the data set. APPEND cannot be specified for members of a partitioned data set.

RECFM(x)
Specifies the record format of the data set. Replace x with either F, V, or U. F specifies that the data set contains fixed length records. V specifies that the data set contains variable length records. U specifies that the data set contains undefined length records. For existing data sets, the default value is the existing record format.

LRECL(n)
Specifies the logical record length (n) for a data set consisting of fixed length records or the maximum logical record length for a data set consisting of variable length records. Values permitted are 1 to 32760. For new data sets, the default value is 80. This parameter is ignored if APPEND is specified. For data sets consisting of variable-length records, the actual logical record length (i.e., the value stored in the Data Set Control Block) following the file transfer, is set to the minimum of the value of LRECL and the longest record in the data set.

BLKSIZE(n)
Specifies the block size (n) of the data set. For data sets containing fixed-length records, the block size must be a multiple of the record length. For data sets containing variable-length records, the block size must be greater than or equal to the record length plus four bytes. The block size must not exceed the track length of the device on which the data set resides. The block size must also not exceed 32,760 bytes. The default value is 3120.

VM/CMS

VM/CMS Send Parameters

Send parameters are optional and are passed as parameters to the host IND$FILE program. The following parameters are valid for the VM/CMS send file transfer operation. When using the following parameters, an open parenthesis character "(" must precede the actual parameter string and a close parenthesis character “)” should follow the parameter string, as in the following example:

(ASCII RECFM F LRECL 125)

ASCII
If the PC file consists of character data only, specify ASCII. This will cause the translation of all graphic and control characters from ASCII to EBCDIC during the transfer so that it is usable on the host computer.

CRLF
Following the convention for ASCII text files, <CR> <LF> pairs are used to terminate records in the PC file, and a CTRL-Z (x'1A') marks the end of file. If the PC file consists of data other than text such as binary format data, do not specify either ASCII or CRLF.

APPEND
If the CMS file already exists and is not empty, APPEND specifies that the contents of the PC file are to be appended to the file.

RECFM x
Specifies the record format of the CMS file. Replace x with either F, V, or U. F specifies that the data set contains fixed length records. V specifies that the data set contains variable length records. U specifies that the data set contains undefined length records. For existing CMS files the default value is the record format of the CMS file.

LRECL n
Specifies the logical record length n for a CMS file consisting of fixed length records or the maximum logical record length for a CMS file consisting of variable length records. Values permitted are 1 to 32760. For new CMS files the default value is 80. This parameter is ignored if APPEND is specified. For CMS files consisting of variable length records, the actual logical record length (i.e., the value stored in the Data Set Control Block) following the file transfer is set to the minimum of the value of LRECL and the longest record in the CMS file.

VM/CMS Receive Parameters

Receive parameters are optional and are passed as parameters to the host IND$FILE program. When using the following parameters, an open parenthesis character "(" must precede the actual parameter string. The close parenthesis character ")" must terminate the parameter string. The following parameters are valid for the VM/CMS receive file transfer operation:

APPEND
APPEND specifies that if the file exists, the CMS file contents are to be appended to it. Unless APPEND is specified, the PC file is replaced by the contents of the CMS file.

ASCII
If the CMS file consists of character data only, specify ASCII. This will cause the translation of all graphic and control characters from ASCII to EBCDIC during the transfer.

CRLF
Following the convention for ASCII text files, <CR> <LF> pairs are used to terminate records in the PC file, and a CTRL-Z (x'1A') marks the end of file.

Keyboard

How to Customize Keyboard

When you create or edit a **Keyboard** profile, you use the **Customize Keyboard Editor**. This tool allows you to assign an action to a keyboard key.

To assign an action to a keyboard key:

1. Highlight the keyboard button you want to customize.
2. Choose the **Normal**, **Shift**, **Ctrl**, **Alt**, **Alt-Gr**, **Alt-Shift**, **Alt-Ctrl**, or **Shift-Ctrl** radio button.
3. Select the **Edit** button.
4. Choose the appropriate action category, then select an item within the category.
5. Click **OK** to return to the **Customize Keyboard Editor** screen.
6. Repeat steps 1-5 as necessary.
7. Click **OK** to save and exit.

Note: to modify an existing or create a new custom keyboard profile on a PASSPORT WEB TO HOST server, follow the steps below:

1. Start the PASSPORT WEB TO HOST Administrator program.
2. Expand the emulation type that you require (3270 Display, 5250 Display, VT Display or Wyse Display) by clicking the plus sign to the left.
3. Choose **Keyboards** under the emulation type.
4. Open an existing keyboard profile from the right pane or create a new one by right-clicking and choosing **New** from the popup menu.
5. Follow steps 1-7 above to assign actions to keys.

Host Key Descriptions

3270 Host Key Descriptions

PASSPORT supports the following key functions:

ALT CURSOR
The Alternate Cursor key toggles the cursor type. There are two cursor types: block and underline.

ATTENTION
Sends a Signal command to the mainframe computer. The Attention key is used as a break key to interrupt the mainframe's execution of the current program. Also, some session manager programs use this key to switch between sessions.

BACK SPACE
This key moves the cursor one character to the left and deletes the current character.

BACK TAB
Moves the cursor back to the first character in an input field or the first character of the previous input field.

CHANGE FORMAT
Toggles the entry assist format screen on and off. While on, allows formatting of tabs and margins. Entry assist must be enabled to use this function.

CLEAR
Erases the display and signals the host system that a clear action occurred.

COLOR xxxx
Changes the text to the selected color.

COLOR INHERIT
Resets the color to the application's default color.

CURSOR BLINK
The cursor is toggled from blinking to non-blinking or from non-blinking to blinking.

CURSOR DOWN
Advances the cursor position down one line on the screen.

CURSOR DOWN 2
Advances the cursor position down two lines on the screen.

CURSOR LEFT
Advances the cursor left one position on the screen.

CURSOR LEFT 2
Advances the cursor left two positions on the screen.

CURSOR RIGHT
Advances the cursor right one position on the screen.

CURSOR RIGHT 2
Advances the cursor right two positions on the screen.

CURSOR RULER
The Cursor Ruler key toggles the cursor ruler (cross-hairs) on and off.

CURSOR SELECT
Indicates that a cursor select action has taken place in the chosen field. The '?' character changes to the '>' character. This key is used in light pen emulation.

CURSOR UP
Advances the cursor up one line on the screen.

CURSOR UP 2
Advances the cursor up two lines on the screen.

DELETE
Deletes the character at the cursor position. All characters on the right of that position are shifted one space to the left.

DUP
Displays the DUP character or an asterisk (*) and the cursor advances to the first character location of the next input field. DUP then provides you with the ability to quickly fill in the information that is the same for every document, such as the date.

END OF FIELD
Moves the cursor to the right of the last character in the same unprotected field.

END OF LINE
Moves the cursor position to the end of the line on the display.

ENTER
Enter key for the host computer. Transmits information to the host.

ENTRY ASSIST
Toggles entry assist mode on and off.

ERASE EOF
Erases the input field from the cursor to the end of the field. The cursor does not move.

ERASE INPUT
Erases all input fields and moves the cursor to the first input character position on the screen.

HIGHLIGHT BLINK
Changes the highlight attribute to blink.

HIGHLIGHT INHERIT
Resets the highlight attribute to the application's default.

HIGHLIGHT REVERSE
Changes the highlight attribute to reverse video.

HIGHLIGHT UNDERSCORE
Changes the highlight attribute to underscore.

HOME
Moves the cursor to the first character of the first input field on the screen.

INSERT
Toggles between insert mode and normal text input mode.

JUMP NEXT
Changes focus to the next available window.

JUMP NEXT ACTIVE
Changes focus to the next active window.

JUMP PREVIOUS
Changes focus to the previous window.

JUMP PREV ACTIVE
Changes focus to the previous active window.

JUMP A - Z
Changes focus to the specified window.

MARK
Displays the field mark character or a semicolon. This key is used when operating with an unformatted display to indicate the end of a field to the program.

NEW LINE
Advances the cursor position to the first input field on the next line.

PA1 - PA3
Program Attention keys.

PF1 - PF24
Program Function keys.

RESET
Recovers from Do Not Enter conditions. This restores the keyboard and turns off the Do Not Enter symbol for all conditions except for Wait. RESET also ends Insert Mode.

SELECT DOWN
Extends or contracts the selection down a row at a time, depending upon which side of the selection bounding box is “anchored.”

SELECT LEFT
Extends or contracts the selection to the left a column at a time, depending upon which side of the selection bounding box is “anchored.”

SELECT RIGHT
Extends or contracts the selection to the right a column at a time, depending upon which side of the selection bounding box is “anchored.”

SELECT UP
Extends or contracts the selection upwards a row at a time, depending upon which side of the selection bounding box is “anchored.”

SYSTEM REQUEST
This key switches your program from a LU-LU session to a SSCP-LU session.

TAB
Moves the cursor to the first character of the next input field.

WORD DELETE
Deletes the word the cursor is positioned on if the word is in an unprotected field.

WORD LEFT
Moves the cursor to the first character of the next word in the field.

WORD LEFT UNP FIELD
Moves the cursor to the first character of the next word in the unprotected field.

WORD RIGHT
Moves the cursor to the first character of the current word or previous word, depending on the location of the cursor.

WORD RIGHT UNP FIELD
Moves the cursor to the first character of the current word or previous word, depending on the location of the cursor in the unprotected field.

WORD WRAP
Toggles Word Wrap on or off. Entry Assist must be enabled to use this feature.

5250 Host Key Descriptions

PASSPORT supports the following key functions:

ALT CURSOR
The Alternate Cursor key sequence toggles the cursor type. There are two cursor types: block and underline.

ATTENTION
Sends a Signal command to the mainframe computer. The Attention key is used as a break key to interrupt the mainframe's execution of the current program. Also, some session manager programs use this key to switch between sessions.

BACK SPACE
This key moves the cursor one character to the left and deletes the current character.

BACK TAB
Moves the cursor back to the first character in an input field or the first character of the previous input field.

CHANGE FORMAT
Toggles the entry assist format screen on and off. While on, allows formatting of tabs and margins. Entry assist must be enabled to use this function.

CLEAR
Erases the display and signals the host system that a clear action occurred.

CURSOR BLINK
Toggles the cursor from blinking to nonblinking or from nonblinking to blinking.

CURSOR DOWN
Advances the cursor position down one line on the screen.

CURSOR DOWN 2
Advances the cursor position down two lines on the screen.

CURSOR LEFT
Advances the cursor left one position on the screen.

CURSOR LEFT 2
Advances the cursor left two positions on the screen.

CURSOR RIGHT
Advances the cursor right one position on the screen.

CURSOR RIGHT 2
Advances the cursor right two positions on the screen.

CURSOR RULER
The Cursor Ruler key toggles the cursor ruler on and off.

CURSOR SELECT
This key is used to indicate that a cursor select action has taken place in the chosen field. The '?' character changes to the '>' character. This key is used in light pen emulation.

CURSOR UP
Advances the cursor up one line on the screen.

CURSOR UP 2
Advances the cursor up two lines on the screen.

DELETE
Deletes the character at the cursor position. All characters on the right of that position are shifted one space to the left.

DUP
Displays the DUP character or an asterisk (*) and the cursor advances to the first character location of the next input field. DUP then provides you with the ability to quickly fill in the information that is the same for every document, such as the date.

END OF FIELD
Moves the cursor to the right of the last character in the same unprotected field.

END OF LINE
Moves the cursor position to the end of the line on the display.

ENTER
Enter key for the host computer. Transmits information to the host.

ENTRY ASSIST
Toggles entry assist mode on and off.

ERASE EOF
Erases the input field from the cursor to the end of the field. The cursor does not move.

ERASE INPUT
This key erases all input fields and moves the cursor to the first input character position on the screen.

FIELD+
The FIELD+ key causes the entry to be positive and advances the cursor to the next field.

FIELD-
The FIELD- key causes the entry to be negative and advances the cursor to the next field.

FIELD EXIT
Causes the cursor to advance to the next field.

HELP
This key provides AS/400 help.

HOME
Moves the cursor to the first character of the first input field on the screen.

INSERT
Toggles between insert mode and normal text input mode.

JUMP NEXT
Changes focus to the next available window.

JUMP NEXT ACTIVE
Changes focus to the next active window.

JUMP PREVIOUS
Changes focus to the previous window.

JUMP PREV ACTIVE
Changes focus to the previous active window.

JUMP A - Z
Changes focus to the specified window.

MARK
Displays the field mark character or a semicolon. This key is used when operating with an unformatted display to indicate the end of a field to the program.

NEW LINE
Advances the cursor position to the first input field on the next line.

PA1 - PA3
Program Attention keys.

PF1 - PF24
Program Function keys.

PRINT
Sends a spooled file to an output queue on the host.

RECORD BACKSPACE
This key backspaces the cursor to the previous record.

RESET
Used to recover from Do Not Enter conditions. This restores the keyboard and turns off the Do Not Enter symbol for all conditions except for Wait. RESET also ends Insert Mode.

ROLL DOWN
Used to roll down to new lines of information on or off the display.

ROLL UP
Used to roll up to new lines of information on or off the display.

SELECT DOWN
Extends or contracts the selection down a row at a time, depending upon which side of the selection bounding box is “anchored.”

SELECT LEFT
Extends or contracts the selection to the left a column at a time, depending upon which side of the selection bounding box is “anchored.”

SELECT RIGHT
Extends or contracts the selection to the right a column at a time, depending upon which side of the selection bounding box is “anchored.”

SELECT UP
Extends or contracts the selection upwards a row at a time, depending upon which side of the selection bounding box is “anchored.”

SYSTEM REQUEST
This key send the 5250 system request key to the host system.

TAB
Moves the cursor to the first character of the next input field.

TEST REQUEST
This key sends a test-key sequence to the AS/400.

WORD DELETE
Deletes the word the cursor is positioned on if the word is in an unprotected field.

WORD LEFT
Moves the cursor to the first character of the next word in the field.

WORD LEFT UNP FIELD
Moves the cursor to the first character of the next word in the unprotected field.

WORD RIGHT
Moves the cursor to the first character of the current word or previous word, depending on the location of the cursor.

WORD RIGHT UNP FIELD
Moves the cursor to the first character of the current word or previous word, depending on the location of the cursor in the unprotected field.

WORD WRAP
Toggles Word Wrap on or off. Entry Assist must be enabled to use this feature.

SCO ANSI Host Key Descriptions

PASSPORT supports the following key functions:

ANSI CENTER
Ansi Center function.

ANSI Ctrl + F1 - F12
Ansi Ctrl F1 - F12 functions.

ANSI Ctrl/Shift + F1 - F12
Ansi Ctrl/Shift F1 - F12 functions.

ANSI DELETE
Ansi Delete function.

ANSI END
Ansi End function.

ANSI F1 - F12
Ansi F1 - F12 functions.

ANSI HOME
Ansi Home function.

ANSI INSERT
Ansi Insert function.

ANSI PAGE DOWN
Ansi Page Down function.

ANSI PAGE UP
Ansi Page Up function.

ANSI Shift + F1 - F12
Ansi Shift F1 - F12 functions.

ALT CURSOR
The key sequence represented is ALT-F11. The Alternate Cursor key toggles the cursor type. There are two cursor types: block and underline.

ANSWER BACK
The key sequence represented is CTRL-F5. The Answer Back key sends an automatic message to the host.

BACK SPACE
This key moves the cursor one character to the left and deletes the current character.

BACK TAB
This key moves the cursor to the first character of the previous tab position.

BREAK
This key sends the Telnet Break command.

COMPOSE
This key sends the VT Compose command.

Ctrl + Sp (NUL)
Null character

Ctrl + A (SOH)
Start of heading. This is a console interrupt.

Ctrl + B (STX)
Start of text

Ctrl + C (ETX)
End of text

Ctrl + D (EOT)
End of transmission

Ctrl + E (ENQ)
Enquiry

Ctrl + F (ACK)
Acknowledge. It clears ENQ.

Ctrl + G (BELL)
Bell. It rings the bell.

Ctrl + H (BS)
Backspace

Ctrl + I (HT)
Horizontal Tab

Ctrl + J (LF)
Line Feed

Ctrl + K (VT)
Vertical tab

Ctrl + L (FF)
Form Feed, page eject

Ctrl + M (CR)
Carriage Return

Ctrl + N (SO)
Shift Out, to an alternate character set

Ctrl + O (SI)
Shift In, to resume the default character set

Ctrl + P (DLE)
Data link escape

Ctrl + Q (DC1)
XON, with XOFF to pause listings, okay to send

Ctrl + R (DC2)
Device control 2, block-mode flow control

Ctrl + S (DC3)
XOFF, with XON is TERM=18 flow control

Ctrl + T (DC4)
Device control 4

Ctrl + U (NAK)
Negative acknowledgement

Ctrl + V (SYN)
Synchronous idle

Ctrl + W (ETB)
End transmission block

Ctrl + X (CAN)
Cancel line

Ctrl + Y (EM)
End of medium interrupt

Ctrl + Z (SUB)
Substitute

Ctrl + [(ESC)
Escape, next character is not echoed.

Ctrl + \ (FS)
File separator

Ctrl +] (GS)
Group separator

Ctrl + ~ (RS)
Record separator, block-mode terminator.

Ctrl + ? (US)
Unit separator

CURSOR BLINK
The cursor is toggled from blinking to non-blinking or from non-blinking to blinking.

CURSOR DOWN
Advances the cursor position down one line on the display.

CURSOR LEFT
Advances the cursor left one position on the screen.

CURSOR RIGHT
Advances the cursor right one position on the screen.

CURSOR RULER
The Cursor Ruler key toggles the cursor ruler on and off.

CURSOR UP
Advances the cursor up one position on the screen.

DELETE
Deletes the character at the cursor position. All characters on the right of that position are shifted one space to the left.

F1-F20
Function keys.

FIND
Moves the cursor to the top left corner of the scrolling region.

HOLD
Toggles the hold state between on and off.

INSERT HERE
Switches between insert and overstrike modes.

JUMP NEXT
Changes focus to the next available window.

JUMP NEXT ACTIVE
Changes focus to the next active window.

JUMP PREVIOUS
Changes focus to the previous window.

JUMP PREV ACTIVE
Changes focus to the previous active window.

JUMP A - Z
Changes focus to the specified window.

LINE FEED
The key sequence represented is SHIFT-ENTER. Moves the cursor position to the next line on the display.

NEXT SCREEN
Displays the next page in page memory on the screen.

NUMPAD x
Sends the appropriate key from the numeric keypad.

PF1 - PF4
Program Function keys.

PREV SCREEN
Displays the previous page in page memory on the screen.

REMOVE
Erases the character on the current page or erases the characters in a field.

RETURN
This is the Return key for the host computer. Transmits information to the host.

SELECT
Switches the terminal between interactive mode and local editing mode.

TAB
Moves the cursor to the first character of the next tab position.

TABS SET
Sets a tab position.

User F6- F20 (UDK's)
Any of the 15 keys (F6 through F20) on the top row of the keyboard for which a user has defined special functions. The host application uses UDKs to store frequently used text and commands.

VT100 and VT220 Host Key Descriptions

PASSPORT supports the following key functions:

ALT CURSOR
The key sequence represented is ALT-F11. The Alternate Cursor key toggles the cursor type. There are two cursor types: block and underline.

ANSWER BACK
The key sequence represented is CTRL-F5. The Answer Back key sends an automatic message to the host.

BACK SPACE
This key moves the cursor one character to the left and deletes the current character.

BACK TAB
This key moves the cursor to the first character of the previous tab position.

BREAK
This key sends the Telnet Break command.

COMPOSE
This key sends the VT Compose command.

Ctrl + ? (US)
Unit separator

Ctrl + [(ESC)
Escape, next character is not echoed.

Ctrl + \ (FS)
File separator

Ctrl +] (GS)
Group separator

Ctrl + ~ (RS)
Record separator, block-mode terminator.

Ctrl + Sp (NUL)
Null character

Ctrl + A (SOH)
Start of heading. This is a console interrupt.

Ctrl + B (STX)
Start of text

Ctrl + C (ETX)
End of text

Ctrl + D (EOT)
End of transmission

Ctrl + E (ENQ)
Enquiry

Ctrl + F (ACK)
Acknowledge. It clears ENQ.

Ctrl + G (BELL)
Bell. It rings the bell.

Ctrl + H (BS)
Backspace

Ctrl + I (HT)
Horizontal Tab

Ctrl + J (LF)
Line Feed

Ctrl + K (VT)
Vertical tab

Ctrl + L (FF)
Form Feed, page eject

Ctrl + M (CR)
Carriage Return

Ctrl + N (SO)
Shift Out, to an alternate character set

Ctrl + O (SI)
Shift In, to resume the default character set

Ctrl + P (DLE)
Data link escape

Ctrl + Q (DC1)
XON, with XOFF to pause listings; :okay to send

Ctrl + R (DC2)
Device control 2, block-mode flow control

Ctrl + S (DC3)
XOFF and XON flow control

Ctrl + T (DC4)
Device control 4

Ctrl + U (NAK)
Negative acknowledgement

Ctrl + V (SYN)
Synchronous idle

Ctrl + W (ETB)
End transmission block

Ctrl + X (CAN)
Cancel line

Ctrl + Y (EM)
End of medium interrupt

Ctrl + Z (SUB)
Substitute

CURSOR BLINK
The key sequence represented is CTRL-F10. The cursor is toggled from blinking to non-blinking or from non-blinking to blinking.

CURSOR DOWN
Advances the cursor position down one line on the display.

CURSOR LEFT
Advances the cursor left one position on the screen.

CURSOR RIGHT
Advances the cursor right one position on the screen.

CURSOR RULER
The Cursor Ruler key toggles the cursor ruler on and off.

CURSOR UP
Advances the cursor up one position on the screen.

DELETE
Deletes the character at the cursor position. All characters on the right of that position are shifted one space to the left.

DO
The VT DO key is the same as the VT F16 key.

F1-F20
Function keys. F15 is the VT HELP key. F16 is the VT DO key. F1 through F5 are local keys.

FIND
Moves the cursor to the top left corner of the scrolling region.

HELP
The VT HELP key is the same as the VT F15 key.

HOLD
Toggles the hold state between on and off.

INSERT HERE
Switches between insert and overstrike modes.

JUMP NEXT
Changes focus to the next available window.

JUMP NEXT ACTIVE
Changes focus to the next active window.

JUMP PREVIOUS
Changes focus to the previous window.

JUMP PREV ACTIVE
Changes focus to the previous active window.

JUMP A - Z
Changes focus to the specified window.

LINE FEED
The key sequence represented is SHIFT-ENTER. Moves the cursor position to the next line on

the display.

NEXT SCREEN
Displays the next page in page memory on the screen.

NUMPAD x
Sends the appropriate key from the numeric keypad.

PF1 - PF4
Program Function keys.

PREV SCREEN
Displays the previous page in page memory on the screen.

REMOVE
Erases the character on the current page or erases the characters in a field.

RETURN
This is the Return key for the host computer. Transmits information to the host.

SELECT
Switches the terminal between interactive mode and local editing mode.

TAB
Moves the cursor to the first character of the next tab position.

TABS SET
Sets a tab position.

User F6- F20 (UDK's)
Any of the 15 keys (F6 through F20) on the top row of the keyboard for which a user has defined special functions. The host application uses UDKs to store frequently used text and commands.

Wyse Host Key Descriptions

PASSPORT supports the following key functions:

ALT CURSOR
The key sequence represented is ALT-F11. The Alternate Cursor key toggles the cursor type. There are two cursor types: block and underline.

BACK SPACE
This key moves the cursor one character to the left.

BREAK
This key sends the Telnet Break command.

CLEAR LINE
Clears the cursor line to space characters, starting at the current cursor position.

CLEAR SCREEN
Clears the page to space characters, starting at the current cursor position

Ctrl + ? (US)
Unit separator

Ctrl + [(ESC)
Escape, next character is not echoed.

Ctrl + \ (FS)
File separator

Ctrl +] (GS)
Group separator

Ctrl + ~ (RS)
Record separator, block-mode terminator.

Ctrl + Sp (NUL)
Null character

Ctrl + A (SOH)
Start of heading. This is a console interrupt.

Ctrl + B (STX)
Start of text

Ctrl + C (ETX)
End of text

Ctrl + D (EOT)
End of transmission

Ctrl + E (ENQ)
Enquiry

Ctrl + F (ACK)
Acknowledge. It clears ENQ.

Ctrl + G (BELL)
Bell. It rings the bell.

Ctrl + H (BS)
Backspace

Ctrl + I (HT)
Horizontal Tab

Ctrl + J (LF)
Line Feed

Ctrl + K (VT)
Vertical tab

Ctrl + L (FF)
Form Feed, page eject

Ctrl + M (CR)
Carriage Return

Ctrl + N (SO)
Shift Out, to an alternate character set

Ctrl + O (SI)
Shift In, to resume the default character set

Ctrl + P (DLE)
Data link escape

Ctrl + Q (DC1)
XON, with XOFF to pause listings; :okay to send

Ctrl + R (DC2)
Device control 2, block-mode flow control

Ctrl + S (DC3)
XOFF and XON flow control

Ctrl + T (DC4)
Device control 4

Ctrl + U (NAK)
Negative acknowledgement

Ctrl + V (SYN)
Synchronous idle

Ctrl + W (ETB)
End transmission block

Ctrl + X (CAN)
Cancel line

Ctrl + Y (EM)
End of medium interrupt

Ctrl + Z (SUB)
Substitute

CURSOR BLINK

The key sequence represented is CTRL-F10. The cursor is toggled from blinking to non-blinking or from non-blinking to blinking.

CURSOR DOWN
Advances the cursor position down one line on the display.

CURSOR LEFT
Advances the cursor left one position on the screen.

CURSOR RIGHT
Advances the cursor right one position on the screen.

CURSOR RULER
The Cursor Ruler key toggles the cursor ruler on and off.

CURSOR UP
Advances the cursor up one position on the screen.

DELETE
Sends ASCII DEL code.

DELETE CHARACTER
Deletes the cursor character, moving all characters to the right of the cursor left one position.

DELETE LINE
Deletes the entire cursor line, moving the lines below it up one line.

ENTER
Moves the cursor to the first position of the current or next line.

ESC
Sends the escape character.

F1-F16
Function keys.

HOME
Moves the cursor to the top left corner of the page.

INSERT
Turns on insert mode.

INSERT CHAR
Inserts a space at the character position, moving all succeeding characters right one position.

INSERT LINE
Inserts a line of space characters below the cursor line.

JUMP NEXT
Changes focus to the next available window.

JUMP NEXT ACTIVE
Changes focus to the next active window.

JUMP PREVIOUS
Changes focus to the previous window.

JUMP PREV ACTIVE
Changes focus to the previous active window.

JUMP A - Z
Changes focus to the specified window.

PAGE NEXT
Displays the next page.

PAGE PREV
Displays the previous page.

PRINT
Sends the formatted page to the printer port.

REPLACE
Turns on replace mode.

SEND
Sends the data from the top of the page through the cursor position to the data port.

Shift+F1 - Shift+F16
Shifted value of function keys.

Shift+HOME
Moves the cursor to the top left corner of the page.

Shift+TAB
Moves cursor to the first character of the previous tab position

TAB
Moves the cursor to the first character of the next tab position.

Macro Scripting Language

Macro Scripting Language Overview

The PASSPORT macro scripting language is a super-set of the Microsoft Visual Basic scripting language. A PASSPORT macro can be used as a starting point for a more complex Visual Basic program.

The complete documentation for Microsoft VBScript can be downloaded from the **Microsoft Script Technologies** page on the MSDN web site.

In addition to the VBScript language commands, there are also many commands and functions used to interface with the host mainframe. All PASSPORT commands utilize the HLLAPI interface to PASSPORT.

When a macro command is executed, PASSPORT runs a separate program, that is actually a HLLAPI application. This HLLAPI application utilizes both the Microsoft VBScript and the PASSPORT HLLAPI module to run the macro.

For PASSPORT WEB TO HOST there are four types of macros that can be used. The first three must be configured by the System Administrator and are not available in PASSPORT PC TO HOST.

- **Group Macros**
- **System Macros**
- **Start Up Macros**
- Local Macros (configured by individual users)

Group macros
Group macros are assigned to all users belonging to a particular group. In order for a group user to have access to these macros, group security must be enabled.

System macros
System macros are available to all sessions. The four types of system macros - 3270 display system macros, 5250 display system macros, VT display system macros and Wyse display system macros, can also be used as Startup macros.

Start Up macros
Start Up macros may be configured (Session→Miscellaneous) to automatically run after a session is established with your host computer. For example, you may want to record a macro that will log on to your host session every time you start PASSPORT.

Note: only **System** macros can be used as **Start Up** macros.

How to Write a Macro

The easiest way to write a macro is to first record a macro using the **Macro->Record** command, and then to modify the macro using the **Macro->Edit** command. Or you can use Notepad or any text editor to edit a macro file. All macro files must have the .zmc file extension.

The structure of the macro file is very simple. The PASSPORT macro must contain a subroutine called **ZMain**.

Macro Skeleton Structure:

Sub ZMain()

Dim Text, ret

.

.

End Sub

Note: The **Dim Text, ret** statement declares two variables that are used in the macro by commands that PASSPORT may record. This statement does not have to be included in your macro if you do not use either of those macro variables.

Macro Command Index

If ... Then ... End If

Sub ... End Sub

Miscellaneous

Beep

Dim

Environ

ExitAllSessions

ExitSession

GetPath

GetVersion

Rem

Shell

Wait Functions

Wait

WaitForCursorPos

WaitForHostUpdate

WaitForNoX

WaitForString

Session Control Functions

Connect

Disconnect

RunConnect

Session Information

Count

Item

ItemHostIP

ItemName

Event Driven Functions

OnScreenChanged

OnHostKey

OnLocalKey

Other VBScript Statements

There are many other VBScript statements that can be used in a PASSPORT macro, including those listed below.

Const

Do...Loop

Erase

Exit

For...Next

For Each...Next

Function

On Error

Option Explicit

Private

Public

Randomize

ReDim

Select Case

Set

While...Wend

Please refer to the Microsoft VBScript documentation for information on these statements if you want to use them.

VBScript Language Documentation

The complete documentation for Microsoft VBScript can be downloaded from the Microsoft Script Technologies page on the MSDN web site.

Cursor Screen and Key Functions

GetCursorColumn

Description

Returns the current column position of the cursor

Syntax

ret = **GetCursorColumn** ( )

Parameters

ret	Integer, output
	The column position of the cursor.

Remarks

The column position ranges in value from 1 to 80 for model 2, 3 and 4 screen sizes. The column position ranges in value from 1 to 132 for the wide column model 5 screen size. For VT screen sizes the column position ranges 1 to 80 or 1 to 132.

The parentheses after **GetCursorColumn** are optional.

This is a PASSPORT function.

Example

```
Sub ZMain()
Dim strText, nRet
Dim nCursorColumn
nCursorColumn = GetCursorColumn
strText = "Current column position = " & nCursorColumn
nRet = MsgBox (strText, 0, "Column Position")
End Sub
```

GetCursorRow

Description

Returns the current row position of the cursor

Syntax

ret = **GetCursorRow** ( )

Parameters

ret	Integer, output
	The row position of the cursor.

Remarks

The row position ranges in value from 1 to 24 for a model 2 screen size, 1 to 32 for a model 3 screen size, 1 to 43 for a model 4 screen size, and 1 to 27 for a model 5 screen size. For VT screen sizes the row position ranges from 1 to 24.

The parentheses after **GetCursorRow** are optional.

This is a PASSPORT function.

Example

```
Sub ZMain()
Dim strText, nRet
Dim nCursorRow
nCursorRow = GetCursorRow
strText = "Current row position = " & nCursorRow
nRet = MsgBox (strText, 0, "Row Position")
End Sub
```

GetOIA

Description

Returns the text in the Operator Information Area (OIA)

Syntax

ret = **GetOIA** ( )

Parameters

ret	String, output
	The text returned from the OIA.

Remarks

The parentheses after **GetOIA** are optional.

This is a PASSPORT function.

Example

```
Sub ZMain()
Dim strText, nRet
Dim nOIA
nOIA = GetOIA
strText = "Text in OIA = " & nOIA
nRet = MsgBox (strText, 0, "OIA Text")
End Sub
```

GetScreenSize

Description

Returns the current screen size

Syntax

ret = **GetScreenSize** ( )

Parameters

ret	Integer, output
	The current screen size in number of character positions (either 1920, 2000, 2560, 3168, 3300, 3440 or 3564)

Remarks

This function returns the actual screen size of the PASSPORT terminal emulator. There are four screen sizes supported: model 2, 3, 4 and 5 plus the VT screen sizes. The screen size determines the number of columns and rows are on the display.

3270 Screen Sizes

1920 = Model 2 (80 columns x 24 rows)

2560 = Model 3 (80 columns x 32 rows)

3440 = Model 4 (80 columns x 43 rows)

3564 = Model 5 (132 columns x 27 rows)

5250 Screen Sizes

1920 = Model 2 (80 columns x 24 rows)

3564 = Model 5 (132 columns x 27 rows)

VT100 and VT220 Screen Sizes

1920 = 80 columns x 24 rows

3168 = 132 columns x 24 rows

SCO-ANSI Screen Sizes

2000 = 80 columns x 25 rows

3300 = 132 columns x 25 rows

The parentheses after **GetScreenSize** are optional.

This is a PASSPORT function.

Example

```
Sub ZMain()
Dim strText, nRet
Dim nScreenSize
nScreenSize = GetScreenSize
strText = "Screen Size = " & nScreenSize
nRet = MsgBox (strText, 0, "Screen Size")
End Sub
```

GetString

Description

Returns a string from the PASSPORT terminal emulation screen

Syntax

ret = **GetString** (row, column, length)

Parameters

ret	String, output
	The string to contain the data read from the PASSPORT screen.
row	Integer, input
	The starting row position in the PASSPORT screen.
column	Integer, input
	The starting column position in the PASSPORT screen.
length	Integer, input
	The number of characters to read.

Remarks

This function reads a string from the PASSPORT terminal emulator screen at the specified position. Row 1 column 1 indicates the home position in the upper left corner of the screen.

In the example, a 10 character string is read from the PASSPORT terminal emulator screen position row 1 column 1 and returned in the text variable called nString.

This is a PASSPORT function.

Example

```
Sub ZMain()
Dim strText, nRet
Dim nString
nString = GetString (1,1,20)
strText = "Text String = " & nString
nRet = MsgBox (strText, 0, "Text String")
End Sub
```

SendEscapeSequence

Description

Sends an escape sequence to the host

Syntax

ret = **SendEscapeSequence** (string)

Parameters

ret	Integer, output
	0= OK
	Non zero = Error
string	String, input
	This string must include a valid ANSI escape sequence.

Remarks

Valid only for VT, SCO-ANSI and Wyse-60 sessions.

This function is similar to the SendHostKeys function, but sends the string as a whole rather than sending each character separately.

VT Host Keys are enclosed within <...>.

This is a PASSPORT function.

Example

```
Sub ZMain()
SendEscapeSequence("<ESC>[C")
End Sub
```

This example sends the escape sequence which moves the cursor one position to the right.

SendHostKeys

Description

Sends ANSI characters and 3270, 5250, VT, SCO-ANSI or Wyse-60 keys to the host mainframe

Syntax

ret = **SendHostKeys** (string)

Parameters

ret	Integer, output
	0= OK
	Non zero = Error
string	String, input
	This string may include both ANSI characters and 3270, 5250, VT, SCO-ANSI or Wyse-60 keys.

Remarks

The string of keys can be any ANSI characters or 3270, 5250, VT, SCO-ANSI or Wyse-60 keys.

This is a PASSPORT function.

3270, 5250, VT, SCO-ANSI or Wyse-60 Keys are enclosed within <...>.

The < character is sent as <<>.

The > character is sent as <>>.

Example

```
Sub ZMain()
SendHostKeys("abc123<TAB>password<ENTER>")
End Sub
```

This example sends the text characters abc123, then the TAB key, then the text characters password and finally the host ENTER key.

Macro Host Key Names for 3270, 5250, VT, SCO-ANSI and Wyse-60

Key Description	Key Value	TN3270	TN5250	VT	SCO-ANSI	Wyse-60
ANSI CENTER	ANSICNTR				X	
ANSI Ctrl + F1	CTRLF1				X	
ANSI Ctrl + F2	CTRLF2				X	
ANSI Ctrl + F3	CTRLF3				X	
ANSI Ctrl + F4	CTRLF4				X	
ANSI Ctrl + F5	CTRLF5				X	
ANSI Ctrl + F6	CTRLF6				X	

ANSI Ctrl + F7	CTRLF7				X	
ANSI Ctrl + F8	CTRLF8				X	
ANSI Ctrl + F9	CTRLF9				X	
ANSI Ctrl + F10	CTRLF10				X	
ANSI Ctrl + F11	CTRLF11				X	
ANSI Ctrl + F12	CTRLF12				X	
ANSI Ctrl/Shift + F1	CTRSFTF1				X	
ANSI Ctrl/Shift + F2	CTRSFTF2				X	
ANSI Ctrl/Shift + F3	CTRSFTF3				X	
ANSI Ctrl/Shift + F4	CTRSFTF4				X	
ANSI Ctrl/Shift + F5	CTRSFTF5				X	
ANSI Ctrl/Shift + F6	CTRSFTF6				X	
ANSI Ctrl/Shift + F7	CTRSFTF7				X	
ANSI Ctrl/Shift + F8	CTRSFTF8				X	
ANSI Ctrl/Shift + F9	CTRSFTF9				X	
ANSI Ctrl/Shift + F10	CTRSFTF10				X	
ANSI Ctrl/Shift + F11	CTRSFTF11				X	
ANSI Ctrl/Shift + F12	CTRSFTF12				X	
ANSI DELETE	ANSIDEL				X	
ANSI END	ANSIEND				X	
ANSI F1	ANSIF1				X	
ANSI F2	ANSIF2				X	
ANSI F3	ANSIF3				X	
ANSI F4	ANSIF4				X	
ANSI F5	ANSIF5				X	
ANSI F6	ANSIF6				X	
ANSI F7	ANSIF7				X	
ANSI F8	ANSIF8				X	
ANSI F9	ANSIF9				X	
ANSI F10	ANSIF10				X	
ANSI F11	ANSIF11				X	
ANSI F12	ANSIF12				X	
ANSI HOME	ANSIHOME				X	
ANSI INSERT	ANSIINS				X	
ANSI PAGE DOWN	ANSIPGDN				X	
ANSI PAGE UP	ANSIPGUP				X	

ANSI Shift + F1	SHIFTF1				X	
ANSI Shift + F2	SHIFTF2				X	
ANSI Shift + F3	SHIFTF3				X	
ANSI Shift + F4	SHIFTF4				X	
ANSI Shift + F5	SHIFTF5				X	
ANSI Shift + F6	SHIFTF6				X	
ANSI Shift + F7	SHIFTF7				X	
ANSI Shift + F8	SHIFTF8				X	
ANSI Shift + F9	SHIFTF9				X	
ANSI Shift + F10	SHIFTF10				X	
ANSI Shift + F11	SHIFTF11				X	
ANSI Shift + F12	SHIFTF12				X	
ALT CURSOR	ALTCURSOR	X	X	X	X	X
ANSWER BACK	ANSWBAK			X	X	
ATTENTION	ATTN	X	X			
BACK SPACE	BACKSPACE	X	X	X	X	X
BACK TAB	BACKTAB	X	X	X	X	
BREAK	BREAK			X	X	X
COMPOSE	COMPOSE			X	X	
CHANGE FORMAT	CHGFMT	X	X			
CLEAR	CLEAR	X	X			
COLOR BLUE	C-BLUE	X				
COLOR GREEN	C-GREEN	X				
COLOR INHERIT	C-INH	X				
COLOR PINK	C-PINK	X				
COLOR RED	C-RED	X				
COLOR TURQUOISE	C-TURQ	X				
COLOR WHITE	C-WHITE	X				
COLOR YELLOW	C-YELLOW	X				
Ctrl + Sp (NUL)	NUL			X	X	X
Ctrl + A (SOH)	SOH			X	X	X
Ctrl + B (STX)	STX			X	X	X
Ctrl + C (ETX)	ETX			X	X	X
Ctrl + D (EOT)	EOT			X	X	X
Ctrl + E (ENQ)	ENQ			X	X	X
Ctrl + F (ACK)	ACK			X	X	X

Ctrl + G (BELL)	BELL			X	X	X
Ctrl + H (BS)	BS			X	X	X
Ctrl + I (HT)	HT			X	X	X
Ctrl + J (LF)	LF			X	X	X
Ctrl + K (VT)	VT			X	X	X
Ctrl + L (FF)	FF			X	X	X
Ctrl + M (CR)	CR			X	X	X
Ctrl + N (SO)	SO			X	X	X
Ctrl + O (SI)	SI			X	X	X
Ctrl + P (DLE)	DLE			X	X	X
Ctrl + Q (DC1)	DC1			X	X	X
Ctrl + R (DC2)	DC2			X	X	X
Ctrl + S (DC3)	DC3			X	X	X
Ctrl + T (DC4)	DC4			X	X	X
Ctrl + U (NAK)	NAK			X	X	X
Ctrl + V (SYN)	SYN			X	X	X
Ctrl + W (ETB)	ETB			X	X	X
Ctrl + X (CAN)	CAN			X	X	X
Ctrl + Y (EM)	EM			X	X	X
Ctrl + Z (SUB)	SUB			X	X	X
Ctrl + [(ESC)	ESC			X	X	X
Ctrl + \\ (FS)	FS			X	X	X
Ctrl +] (GS)	GS			X	X	X
Ctrl + ~ (RS)	RS			X	X	X
Ctrl + ? (US)	US			X	X	X
CURSOR BLINK	CURBLINK	X	X	X	X	X
CURSOR DOWN	DOWN	X	X	X	X	X
CURSOR DOWN 2	DOWN2	X	X			
CURSOR LEFT	LEFT	X	X	X	X	X
CURSOR LEFT 2	LEFT2	X	X			
CURSOR RIGHT	RIGHT	X	X	X	X	X
CURSOR RIGHT 2	RIGHT2	X	X			
CURSOR RULER	RULER	X	X	X	X	X
CURSOR SELECT	CRSRSEL	X	X			
CURSOR UP	UP	X	X	X	X	X
CURSOR UP 2	UP2	X	X			

DELETE	DELETE	X	X	X	X	X
DUP	DUP	X	X			
END OF FIELD	ENDFIELD	X	X			
END OF LINE	ENDOFLINE	X	X			
ENTER	ENTER	X	X			X
ENTRY ASSIST	ENTRYASSIST	X	X			
ERASE EOF	ERASEEOF	X	X			
ERASE INPUT	ERASEINPU	X	X			
F1	VTF1			X	X	
F2	VTF2			X	X	
F3	VTF3			X	X	
F4	VTF4			X	X	
F5	VTF5			X	X	
F6	PF6			X	X	
F7	PF7			X	X	
F8	PF8			X	X	
F9	PF9			X	X	
F10	PF10			X	X	
F11	PF11			X	X	
F12	PF12			X	X	
F13	PF13			X	X	
F14	PF14			X	X	
F15-HELP	PF15			X	X	
F16-DO	PF16			X	X	
F17	PF17			X	X	
F18	PF18			X	X	
F19	PF19			X	X	
F20	PF20			X	X	
FIELD +	FIELD+		X			
FIELD -	FIELD-		X			
FIELD EXIT	FIELDEXIT		X			
FIND	FIND			X	X	
HELP	HELP		X			
HIGHLIGHT BLINK	H-BLINK	X				
HIGHLIGHT INHERIT	H-INH	X				
HIGHLIGHT REVERSE	H-REV	X				

HIGHLIGHT UNDERSCORE	H-UNDER	X				
HOLD	HOLD					
HOME	HOME	X	X			X
INSERT	INSERT	X	X			X
INSERT HERE	INSERTHERE			X	X	
JUMP NEXT	JUMPNEXT	X	X	X	X	X
JUMP NEXT ACTIVE	JUMPNEXTA	X	X	X	X	X
JUMP PREVIOUS	JUMPPREV	X	X	X	X	X
JUMP PREV ACTIVE	JUMPPREVA	X	X	X	X	X
JUMP A	JUMPA	X	X	X	X	X
JUMP B	JUMPB	X	X	X	X	X
JUMP C	JUMPC	X	X	X	X	X
JUMP D	JUMPD	X	X	X	X	X
JUMP E	JUMPE	X	X	X	X	X
JUMP F	JUMPF	X	X	X	X	X
JUMP G	JUMPG	X	X	X	X	X
JUMP H	JUMPH	X	X	X	X	X
JUMP I	JUMPI	X	X	X	X	X
JUMP J	JUMPJ	X	X	X	X	X
JUMP K	JUMPK	X	X	X	X	X
JUMP L	JUMPL	X	X	X	X	X
JUMP M	JUMPM	X	X	X	X	X
JUMP N	JUMPN	X	X	X	X	X
JUMP O	JUMPO	X	X	X	X	X
JUMP P	JUMPP	X	X	X	X	X
JUMP Q	JUMPQ	X	X	X	X	X
JUMP R	JUMPR	X	X	X	X	X
JUMP S	JUMPS	X	X	X	X	X
JUMP T	JUMPT	X	X	X	X	X
JUMP U	JUMPU	X	X	X	X	X
JUMP V	JUMPV	X	X	X	X	X
JUMP W	JUMPW	X	X	X	X	X
JUMP X	JUMPX	X	X	X	X	X
JUMP Y	JUMPY	X	X	X	X	X
JUMP Z	JUMPZ	X	X	X	X	X

LINE FEED	LINEFEED			X	X	
MARK	MARK	X	X			
NEW LINE	NEWLINE	X	X			
NEXT SCREEN	NEXTSCR			X	X	
NUMPAD ,	NPCOMMA			X	X	
NUMPAD -	NPMINUS			X	X	
NUMPAD .	NPPERIOD			X	X	
NUMPAD 0	NP0			X	X	
NUMPAD 1	NP1			X	X	
NUMPAD 2	NP2			X	X	
NUMPAD 3	NP3			X	X	
NUMPAD 4	NP4			X	X	
NUMPAD 5	NP5			X	X	
NUMPAD 6	NP6			X	X	
NUMPAD 7	NP7			X	X	
NUMPAD 8	NP8			X	X	
NUMPAD 9	NP9			X	X	
NUMPAD ENTER	NPENTER			X	X	
PA1	PA1	X	X			
PA2	PA2	X	X			
PA3	PA3	X	X			
PF1	PF1	X	X	X	X	
PF2	PF2	X	X	X	X	
PF3	PF3	X	X	X	X	
PF4	PF4	X	X	X	X	
PF5	PF5	X	X			
PF6	PF6	X	X			
PF7	PF7	X	X			
PF8	PF8	X	X			
PF9	PF9	X	X			
PF10	PF10	X	X			
PF11	PF11	X	X			
PF12	PF12	X	X			
PF13	PF13	X	X			
PF14	PF14	X	X			
PF15	PF15	X	X			

PF16	PF16	X	X			
PF17	PF17	X	X			
PF18	PF18	X	X			
PF19	PF19	X	X			
PF20	PF20	X	X			
PF21	PF21	X	X			
PF22	PF22	X	X			
PF23	PF23	X	X			
PF24	PF24	X	X			
PREV SCREEN	PREVSCR			X	X	
PRINT	PRINT		X			X
RECORD BACKSPACE	RECBACKSPACE		X			
REMOVE	REMOVE			X	X	
RESET	RESET	X	X			
RETURN	RETURN			X	X	
ROLL DOWN	ROLLDOWN		X			
ROLL UP	ROLLUP		X			
SELECT	SELECT			X	X	
SELECT DOWN	SELDN	X	X			
SELECT LEFT	SELLT	X	X			
SELECT RIGHT	SELRT	X	X			
SELECT UP	SELUP	X	X			
SYSTEM REQUEST	SYSREQ	X	X			
TAB	TAB	X	X	X	X	X
TABS SET	TABSSET			X	X	
TEST REQUEST	TESTREQ		X			
USER F6	USERPF6			X	X	
USER F7	USERPF7			X	X	
USER F8	USERPF8			X	X	
USER F9	USERPF9			X	X	
USER F10	USERPF10			X	X	
USER F11	USERPF11			X	X	
USER F12	USERPF12			X	X	
USER F13	USERPF13			X	X	
USER F14	USERPF14			X	X	

USER F15	USERPF15			X	X	
USER F16	USERPF16			X	X	
USER F17	USERPF17			X	X	
USER F18	USERPF18			X	X	
USER F19	USERPF19			X	X	
USER F20	USERPF20			X	X	
WORD DELETE	WORDDEL	X	X			
WORD LEFT	WORDL	X	X			
WORD LEFT UNP FIELD	WORDLUNP	X	X			
WORD RIGHT	WORDR	X	X			
WORD RIGHT UNP FIELD	WORDRUNP	X	X			
WORD WRAP	WORDWRAP	X	X			
WYSE F1	WF1					X
WYSE F2	WF2					X
WYSE F3	WF3					X
WYSE F4	WF4					X
WYSE F5	WF5					X
WYSE F6	WF6					X
WYSE F7	WF7					X
WYSE F8	WF8					X
WYSE F9	WF9					X
WYSE F10	WF10					X
WYSE F11	WF11					X
WYSE F12	WF12					X
WYSE F13	WF13					X
WYSE F14	WF14					X
WYSE F15	WF15					X
WYSE F16	WF16					X
WYSE PAGE NEXT	WPGNEXT					X
WYSE PAGE PREV	WPGPREV					X
WYSE Shift + F1	WSFTF1					X
WYSE Shift + F2	WSFTF2					X
WYSE Shift + F3	WSFTF3					X
WYSE Shift + F4	WSFTF4					X
WYSE Shift + F5	WSFTF5					X

WYSE Shift + F6	WSFTF6					X
WYSE Shift + F7	WSFTF7					X
WYSE Shift + F8	WSFTF8					X
WYSE Shift + F9	WSFTF9					X
WYSE Shift + F10	WSFTF10					X
WYSE Shift + F11	WSFTF11					X
WYSE Shift + F12	WSFTF12					X
WYSE Shift + F13	WSFTF13					X
WYSE Shift + F14	WSFTF14					X
WYSE Shift + F15	WSFTF15					X
WYSE Shift + F16	WSFTF16					X
WYSE CLEAR LINE	WCLRLINE					X
WYSE CLEAR SCREEN	WCLRSCRN					X
WYSE DELETE CHAR	WDELCHAR					X
WYSE DELETE LINE	WDELLINE					X
WYSE Shift + HOME	WSFTHOME					X
WYSE INSERT CHAR	WINSCHAR					X
WYSE INSERT LINE	WINSLINE					X
WYSE REPLACE	WREPL					X
WYSE SEND	WSEND					X
WYSE Shift + TAB	WSFTTAB					X
WYSE ESC	WESC					X

SendString

Description

Copies a string into the PASSPORT terminal emulation screen, to later be sent back to the host mainframe

Syntax

ret = **SendString** (row, column, string)

Parameters

ret	Integer, output
	0 = OK
	Non Zero = Error
row	Integer, Input
	The starting row position in the PASSPORT screen.
column	Integer, Input
	The starting column position in the PASSPORT screen.
string	String, Input
	The string to copy into the PASSPORT terminal emulation screen.

Remarks

This function copies a string into the PASSPORT terminal emulator screen at the specified row and column position. Row 1 column 1 indicates the home position in the upper left corner of the screen.

Strings should only be copied into unprotected fields. When copied into an unprotected field, the MDT (Modified Data Tag) is set in that field so that the data is transmitted to the host whenever the ENTER key or other inbound AID key is sent.

This is a PASSPORT function.

Example

```
Sub ZMain()
SendString 20,3,"abc123"
End Sub
```

This example copies the string abc123 into the PASSPORT terminal emulation screen starting at row 20 column 3.

SetCursor

Description

Position the cursor at the new position

Syntax

ret = **SetCursor** (row, column)

Parameters

ret	Integer, output
	0 = OK
	Non zero = Error
row	Integer, input
	The new row position of the cursor.
column	Integer, input
	The new column position of the cursor.

Remarks

This function moves the cursor to a new position specified by a row and column position. Row 1 column 1 indicates the home position in the upper left corner of the screen. Care must be taken not to position the cursor off the screen.

This is a PASSPORT function.

Example

```
Sub ZMain()
SetCursor 1,1
End Sub
```

This example positions the cursor at the home position, row 1 column 1.

Dialog Box Functions

MsgBox

Description

Displays a message in a dialog box and waits for the user to choose a button

Syntax

ret = **MsgBox** (MsgString, type, TitleString)

Parameters

ret	Integer, output
	Indicates which button the user pressed.
	1,2,3,4,6,7 = OK, Cancel, Abort, Retry, Yes, No
MsgString	String, input
	This is the message displayed in the dialog box.
type	Integer, input
	Indicates the number and type of buttons, and other attributes of the dialog box. The value is the sum of the following numbers, one from each group.
	Group 1 – Number and type of Buttons
	0 = OK
	1 = OK/Cancel
	2 = Abort/Retry/Ignore
	3 = Yes/No/Cancel
	4 = Yes/No
	5 = Retry/Cancel
	Group 2 – Type of Icon
	16 = Stop icon
	32 = Question mark icon
	48 = Exclamation mark icon
	64 = Information mark icon
	Group 3 – Button Default
	0 = 1st button is default
	256 = 2nd button is default
	512 = 3rd button is default
	Group 4 – Type of Dialog Box
	0 = Application modal
	4096 = System modal
TitleString	String, input
	The text string that appears as the title of the dialog box.

Remarks

The MsgBox function returns a value indicating which button the user has chosen. Refer to the Microsoft VBScript documentation for additional information.

This is a VBScript function.

Example

```
Sub ZMain()
TitleString="PASSPORT Macro"
MsgBox "Do you want to Continue?",4,TitleString
End Sub
```

MsgBoxGetInput

Description

Displays a message box with an input field

Syntax

ret = **MsgBoxGetInput** (MessageString)

Parameters

ret	String, output
	The string that is entered into the dialog box by the user.
MessageString	String, input
	The string that is displayed as the message in the dialog box.

Remarks

This is a PASSPORT function.

Example

```
Sub ZMain()
strName=MsgBoxGetInput("Please enter your first name:")
MsgBox "Hello, "&strName
End Sub
```

MsgBoxGetPassword

Description

Displays a message box with a password non-display input field

Syntax

ret = **MsgBoxGetPassword** (MessageString)

Parameters

ret	String, output
	The string that is entered into the dialog box by the user.
MessageString	String, input
	The string that is displayed as the message in the dialog box.

Remarks

As the user enters the text, the asterisks character is displayed instead of the actual character that is typed.

This is a PASSPORT function.

Example

```
Sub ZMain()
strPassword=MsgBoxGetPassword("Please enter your password:")
MsgBox "Your password is: "&strPassword
End Sub
```

File, Print and Directory Functions

AppendPSToFile

Description

Copies the area from the PASSPORT presentation space and appends it to a file.

Syntax

ret = **AppendPSToFile** (FileName, StartRow, StartColumn, EndRow, EndColumn)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
FileName	String, input
	The name of the file that the text is appended to.
StartRow	Integer, input
	The starting row position in the PASSPORT screen.
StartColumn	Integer, input
	The starting column position in the PASSPORT screen.
EndRow	Integer, input
	The ending row position in the PASSPORT screen.
EndColumn	Integer,Input
	The ending column position in the PASSPORT screen.

Remarks

The PASSPORT presentation space is the same as the PASSPORT terminal emulation screen area.

This is a PASSPORT function.

Example

```
Sub ZMain()
AppendPSToFile "C:/Test.txt",1,1,24,80
End Sub
```

ChDir

Description

Changes the current directory.

Syntax

ret = **ChDir** (DirString)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
DirString	String, input
	Directory name of the new directory.

Remarks

This is a PASSPORT command. This command is a Visual Basic command but is not implemented by VBScript.

Example

```
Sub ZMain()
ChDir "c:\temp"
End Sub
```

ChDrive

Description

Changes the current drive.

Syntax

ret = **ChDrive** (DriveString)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
DriveString	String, input
	The string that contains the drive letter to change to.

Remarks

This is a PASSPORT command. This command is a Visual Basic command but is not implemented by VBScript.

Example

```
Sub ZMain()
ChDrive "A"
End Sub
```

CurDir

Description

This command returns a string that is the current directory.

Syntax

ret = **CurDir** ( )

Parameters

ret	String,output
	The current directory is returned in this string.

Remarks

When the macro starts execution the PASSPORT working directory is the current directory.

Parentheses are optional for the CurDir command.

This is a PASSPORT command. This command is a Visual Basic command but is not implemented by VBScript.

Example

```
Sub ZMain()
strDir=CurDir
MsgBox "The current directory is: " & strDir
End Sub
```

FTPReceiveFile

Description

Transfers a file from the host to the PC using FTP file transfer

Syntax

ret = **FTPReceiveFile** (LocalFile, HostFile)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
LocalFile	String, input
	This string contains the path and name of the local file to transfer.
HostFile	String, input
	This string contains the name of the file to create on the host.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
ret = FTPReceiveFile ("C:\Program Files\MyFolder\LocalFile.txt",
"HostFile.txt")
End Sub
```

FTPSendFile

Description

Transfers a file from the PC to the host using FTP file transfer

Syntax

ret = **FTPSendFile** (LocalFile, HostFile)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
LocalFile	String, input
	This string contains the path and name of the local file to transfer.
HostFile	String, input
	This string contains the name of the file to create on the host.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
ret = FTPSendFile ("C:\Program Files\MyFolder\LocalFile.txt",
"HostFile.txt")
End Sub
```

Kill

Description

The kill command erases or deletes a file or group of files.

Syntax

ret = **Kill** (FileString)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
FileString	String, input
	This is the file name or file name to delete.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
Kill "c:\test.txt"
End Sub
```

MkDir

Description

Creates a new directory.

Syntax

ret = **MkDir** (DirString)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
DirString	String, input
	The string that contains the name of the directory to make.

Remarks

This is a PASSPORT command. This command is a Visual Basic command but is not implemented by VBScript.

Example

```
Sub ZMain()
MkDir "C:\Temp"
End Sub
```

PrintFile

Description

Prints the file using the printer that is defined in the PASSPORT display session.

Syntax

ret = **PrintFile** (FileName)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
FileName	String, input
	This string contains the name of the file to print.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
PrintFile "C:\Temp\Test.txt"
End Sub
```

PrintPS

Description

Prints the specified area of the PASSPORT presentation space to the printer defined in the PASSPORT display session.

Syntax

ret = **PrintPS** (StartRow, StartColumn, EndRow, EndColumn)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
StartRow	Integer, input
	The starting row position in the PASSPORT screen.
StartColumn	Integer, input
	The starting column position in the PASSPORT screen.
EndRow	Integer, input
	The ending row position in the PASSPORT screen.
EndColumn	Integer,Input
	The ending column position in the PASSPORT screen.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
PrintPS 1,1,24,80
End Sub
```

ReceiveFile

Description

Receives a file from the host to the PC.

Syntax

ret = **ReceiveFile** (String)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
String	String, input
	The string that contains the receive file names and options.

Remarks

This is a PASSPORT command.

If the PC file name contains spaces, it must be enclosed in double quotes. For example: ReceiveFile (""""C:\Documents and Settings\My Profile\Desktop\test.txt"" README.TXT ASCII CRLF")

Example

```
Sub ZMain()
ReceiveFile "c:\pc.txt 'host.txt' ascii crlf"
End Sub
```

RmDir

Description

Removes a directory.

Syntax

ret = **RmDir** (DirString)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
DirString	String, input
	This string contains the name of the directory to be removed.

Remarks

This is a PASSPORT command. This command is a Visual Basic command but is not implemented by VBScript.

Note: The target directory must be empty for this command to work without returning an error.

Example

```
Sub ZMain()
RmDir "C:\Temp"
End Sub
```

SendFile

Description

Sends a file from the PC to the host using IND$FILE

Syntax

ret = **SendFile** (String)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
String	String, input
	This string contains the send file names and options.

Remarks

This is a PASSPORT command.

If the PC file name contains spaces, it must be enclosed in double quotes. For example: SendFile (""""C:\Documents and Settings\My Profile\Desktop\test.txt"" README.TXT ASCII CRLF")

Example

```
Sub ZMain()
SendFile "c:\pc.txt 'host.txt' ascii crlf"
End Sub
```

WritePSToFile

Description

Copies the area from the PASSPORT presentation space and writes it to a file

Syntax

ret = **WritePSToFile** (FileName, StartRow, StartColumn, EndRow, EndColumn)

Parameters

ret	Integer,output
	0 = OK
	Non zero = Error
FileName	String, input
	The file the text is written to.
StartRow	Integer, input
	The starting row position in the PASSPORT screen.
StartColumn	Integer, input
	The starting column position in the PASSPORT screen.
EndRow	Integer, input
	The ending row position in the PASSPORT screen.
EndColumn	Integer,Input
	The ending column position in the PASSPORT screen.

Remarks

The PASSPORT presentation space is the same as the PASSPORT terminal emulation screen area.

Note: If the target file exists, it will be overwritten.

This is a PASSPORT command.

Example

```
Sub ZMain()
WritePSToFile "c:\test.txt",1,1,24,80
End Sub
```

Flow of Control Statements

Call

Description

Transfers control to a Sub procedure or Function procedure. This is a VBScript statement and is fully documented in the Microsoft VBScript documentation.

Syntax

[**Call**] Name [Argumentlist]

Parameters

Call	Optional keyword
	If specified, you must enclose argumentlist in parentheses.
	For example: Call MyProc (0)
Name	Required
	Name of the procedure to call.
Agrumentlist	Optional
	This is a comma-delimited list of variables, arrays, or expressions to pass to the procedure.

Remarks

Transfers program control to a sub procedure. After the sub procedure is executed, the statement after the Call statement is executed next. The sub procedure is defined using the Sub and End Sub statements.

This is a VBScript command. For additional information refer to the Microsoft VBScript documentation.

If ... Then ... End If

Description

Conditionally executes a group of statements, depending on the value of an expression. These are VBScript statements and are fully documented in the Microsoft VBScript documentation.

Syntax

If condition **Then**

[statements]

[**ElseIf** condition-n **Then**

[statements-n]] . . .

[**Else**

[elsestatements]]

End If

Parameters

condition	Can be one or more of the following expressions:
	A numeric or string expression is either True or False. If the condition is Null, then the condition is treated as False.
statements	Statements separated by colons; executed if condition is True.
condition-n	Same as condition.
statements-n	Statements executed if the associated condition-n is True.
elsestatements	Statements executed if no previous condition or condition-n expression is True.

Remarks

The condition can be either a comparison between two strings or two integers. A comparison between a string and an integer is invalid. The comparison operators for a string comparison are = and <>. The comparison operators for an integer comparison are <, <=, >, >=, =, and <>.

This is a VBScript command. For additional information refer to the Microsoft VBScript documentation.

Sub ... End Sub

Description

Declares the name, arguments, and code that form the body of a Sub procedure. These are VBScript statements and are fully documented in the Microsoft VBScript documentation.

Syntax

Sub name [(arglist)]

[statements]

[**Exit Sub**]

[statements]

End Sub

Parameters

name	Required
	The name of the sub procedure.
arglist	Optional
	List of variables or expressions.
statements	Valid VBScript statements.

Remarks

All executable code must be contained inside of procedures. You cannot define a Sub procedure inside another Sub procedure. The Exit Sub statement causes an immediate exit from a Sub procedure.

This is a VBScript command. For additional information refer to the Microsoft VBScript documentation.

Example

```
Sub OutputMessage

.

.

End Sub
```

Miscellaneous

Beep

Description

Sounds a tone through the computer's speaker

Syntax

Beep ( )

Parameters

None

Remarks

Parentheses are optional. This is a PASSPORT command.

Example

```
Sub ZMain()
Beep
End Sub
```

Dim

Description

Declares variables and allocates storage space

Syntax

Dim variablename

Parameters

variablename	A string of characters indicating the name of the variable.

Remarks

This is a Microsoft VBScript statement. For additional information refer to the Microsoft VBScript documentation.

Example

```
Dim Count

Dim FileName
```

Environ

Description

Returns the string associated with an operating system environment variable

Syntax

ret = **Environ** (envstring)

Parameters

ret	String, output
	This is the actual string associated with the environment string name.
envstring	String, input
	This is the name of the environment string.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
strEnv=Environ("PASSPORT")
MsgBox "The variable is: " & strEnv
End Sub
```

If the SET PASSPORT=C:\PASSPORT is in the AUTOEXEC.BAT file, the string would have the value “C:\PASSPORT”.

ExitAllSessions

Description

Ends the macro and forces all PASSPORT sessions to close

Syntax

ExitAllSessions ( )

Parameters

None

Remarks

This is the same as using the **File→Exit All** (PC TO HOST) or **File→Close All** (WEB TO HOST) command from the menu. It is recommended that the user be at the mainframe logon screen when this macro command is executed.

Parentheses are optional. This is a PASSPORT command.

Example

```
Sub ZMain()
ExitAllSessions
End Sub
```

ExitSession

Description

Ends the macro and forces the session running this macro to close

Syntax

ExitSession ( )

Parameters

None

Remarks

This is the same as using the **File→Exit** (PC TO HOST) or **File→Close** (WEB TO HOST) command from the menu. It is recommended that the user be at the mainframe logon screen when this macro command is executed.

Parentheses are optional. This is a PASSPORT command.

Example

```
Sub ZMain()
ExitSession
End Sub
```

GetPath

Description

Gets the path of the PASSPORT software working directory.

Syntax

ret = **GetPath** ( )

Parameters

ret	String, output
	This string returns the PASSPORT working directory.

Remarks

Parentheses are optional. This is a PASSPORT command. This command is not valid for PASSPORT WEB TO HOST and will return a null value if used.

Example

```
Sub ZMain()
strPath=GetPath()
MsgBox "The current path is: " & strPath
End Sub
```

GetVersion

Description

Gets the version number of the PASSPORT software

Syntax

ret = **GetVersion** ( )

Parameters

ret	String, output
	The string to receive the version number, such as “Version 2004-930”.

Remarks

Parentheses are optional. This is a PASSPORT command.

Example

```
Sub ZMain()
strVer=GetVersion()
MsgBox strVer
End Sub
```

Rem

Description

This command indicates that this statement is a remark and is just a comment line

Syntax

Rem string

Parameters

string	Any text you wish to be used for comments in your macro.

Remarks

Also the single quote character ‘ can be used to indicate a comment line. This is a VBScript command.

Examples

```
Rem This is a comment line

' This is another comment line
```

Shell

Description

Runs an executable Windows program

Syntax

ret = **Shell** (commandstring, style)

Parameters

ret	Integer, output
	0 = OK
	Non zero = Error
commandstring	String, input
	This is the name of the program to execute along with any command line parameters.
style	Integer, input
	This parameter determines window style and focus of the program's window when it is executed and initially displayed.
	1,5,9 = normal with focus 2 = minimized with focus 3 = maximized with focus 4,8 = normal without focus 6,7 = minimized without focus

Remarks

This is a VBScript command.

Example

```
Sub ZMain()
ret = Shell ("notepad.exe", 3)
End Sub
```

Wait Functions

Wait

Description

Waits for the specified number of seconds and then returns

Syntax

Wait (time)

Parameters

time	Integer, input
	The number of seconds to wait.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
Wait(10)
MsgBox "Ten seconds has expired."
End Sub
```

WaitForCursorPos

Description

Waits until the host has positioned the cursor at a certain row and column position

Syntax

ret = **WaitForCursorPos** (row, column, timeout)

Parameters

ret	Integer, output
	0 = OK
	Non zero = Timed Out
row	Integer, input
	The row position to wait for the cursor to be positioned at.
column	Integer, input
	The column position to wait for the cursor to be positioned at.
timeout	Integer, input
	The number of seconds to wait before indicating a time out condition.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
WaitForCursorPos 1,1,10
End Sub
```

WaitForHostUpdate

Description

Waits until a host screen update has been performed

Syntax

ret = **WaitForHostUpdate** (timeout)

Parameters

ret	Integer, output
	0 = OK
	Non zero = Timed Out
timeout	Integer, input
	The number of seconds to wait before indicating a time out condition.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
WaitForHostUpdate 10
End Sub
```

WaitForNoX

Description

Waits until the X () or X SYSTEM or other X indicator has been removed from the Operator Information Area (OIA) status line.

Syntax

ret = **WaitForNoX** (timeout)

Parameters

ret	Integer, output
	0 = OK
	Non zero = Timed Out
timeout	Integer, input
	The number of seconds to wait before indicating a time out condition.

Remarks

If the X is still present on the OIA after the number of seconds specified, the function will return with the return code indicating a timeout error condition.

This is a PASSPORT command.

Example

```
Sub ZMain()
WaitForNoX 10
End Sub
```

WaitForString

Description

Waits until the specified string is present at the specified row and column position

Syntax

ret = **WaitForString** (row, column, string, timeout)

Parameters

ret	Integer, output
	0 = OK
	Non zero = Timed Out
row	Integer, input
	The row position to look for the specified string.
column	Integer, input
	The column position to look for the specified string.
string	String, input
	The string to wait for at the specified row and column position.
timeout	Integer, input
	The number of seconds to wait before indicating a time out condition.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
WaitForString 21,32,"logon",10
End Sub
```

Session Control Functions

Connect

Description

Establishes a connection with the host.

Syntax

ret = **Connect** (timeout)

Parameters

ret	Integer, output
	0 = OK
	Non zero = timed out/error
timeout	Integer, input
	The number of seconds to wait before indicating a time out condition.

Remarks

This is a PASSPORT function.

Example

```
Sub ZMain()
Disconnect
msgbox ("The session is now disconnected!")
Connect 10
msgbox ("The session is now connected!")
End Sub
```

Disconnect

Description

Ends the connection with the host.

Syntax

ret = **Disconnect** ( )

Parameters

ret	Integer, output
	0 = OK
	Non zero = error

Remarks

This is a PASSPORT function.

Example

```
Sub ZMain()
Disconnect
msgbox ("The session is now disconnected!")
Connect 10
msgbox ("The session is now connected!")
End Sub
```

RunConnect

Description

Opens the specified session and establishes a connection with the host.

Syntax

ret = **RunConnect** (sessionname, timeout, visible)

Parameters

ret	String, output
	The HLLAPI short name which ranges from uppercase 'A' to 'Z'.
sessionname	String, input
	Name of session to open, including full path.
timeout	Integer, input
	The number of seconds to wait before indicating a time out condition.
visible	Integer, input
	0 = Not visible 1 = Visible

Remarks

This is a PASSPORT function.

For PASSPORT PC TO HOST, the sessionname parameter should specify path and file name of the session profile to open such as "C:\Program Files\PASSPORT\Passport.zws".

For PASSPORT WEB TO HOST, the sessionname parameter must include the URL for the session profile such as "http://servername/pec/Ecomes.asp?sessionprofile=3270dsp/Sessions/passport".

Example

```
Sub ZMain()
RunConnect "c:/program files/passport/passport.zws", 10, 1
End Sub
```

Session Information

Count

Description

This is a property and not a method. This property contains the number of PASSPORT WEB TO HOST and/or PASSPORT PC TO HOST sessions that are currently open and connected to a host session.

Syntax

ret = **Count**

Parameters

ret	Integer, output

Remarks

This is a property of the PASSMAC.DLL object. This property is provided by PASSPORT and is not part of VBScript.

Example

```
Sub ZMain()
Dim strText, nRet
Dim nSessionCount
nSessionCount = Count
strText = "Number of Sessions = " & nSessionCount
nRet = MsgBox (strText, 0, "Number of Sessions")
End Sub
```

Item

Description

This method returns the HLLAPI short name for a specific session.

Syntax

ret = **Item** (n)

Parameters

ret	String, output
	The HLLAPI short name which ranges from uppercase 'A' to 'Z'.
n	Ranges in value from 1 to the maximum number of PASSPORT sessions that are connected.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
Dim strText, nRet
Dim strShortName
strShortName = Item(1)
strText = "Session Short Name = " & strShortName
nRet = MsgBox (strText, 0, "Short Name")
End Sub
```

ItemHostIP

Description

This method returns the IP Address of the connected host.

Syntax

ret = **ItemHostIP** (n)

Parameters

ret	String, output
	The host IP address.
n	Ranges in value from 1 to the maximum number of PASSPORT sessions that are connected.

Remarks

This is a PASSPORT command.

Example

```
Sub ZMain()
Dim strText, nRet
Dim strHostIP
strHostIP = ItemHostIP (1)
strText = "Host IP Address = " &strHostIP
nRet = MsgBox (strText, 0, "Host IP Address")
End Sub
```

ItemName

Description

This method gets the session profile name for a specific session. The session profile name is the name of the session that was launched.

Syntax

ret = **ItemName** (n)

Parameters

ret	String, output
	The session profile name.
n	Ranges in value from 1 to the maximum number of PASSPORT sessions that are connected.

Remarks

Note: It is possible to have multiple sessions open with the same session profile name. However, it is not possible to have multiple sessions with the same HLLAPI short name.

Example

```
Sub ZMain()
Dim strText, nRet
Dim strSessionName
strSessionName = ItemName (1)
strText = "Session Profile Name = " &strSessionName
nRet = MsgBox (strText, 0, "Session Name")
End Sub
```

Event Driven Functions

OnScreenChanged

Description

This is an event driven function. This function is called whenever the PASSPORT host screen has changed. If is called whenever the screen is updated by the host. This function is not called when the screen is updated by the user.

Syntax

Sub **OnScreenChanged** (nState, nStart, nEnd)
End Sub

Parameters

nState	Integer, input
	Always 0, reserved for future use
nStart	Integer, input
	Always 0, reserved for future use
nEnd	Integer, input
	Always 0, reserved for future use

Remarks

Zephyr does not recommend using event driven functions inside of a PASSPORT macro. Instead, these functions should be used in an external program or web page that uses the PASSMAC.DLL object to interface with the PASSPORT terminal emulator program. Below is an example of using this function inside of a web page using VBScript.

Example

```
<HTML>

<OBJECT name=passmac classid="CLSID:6440DEE3-7072-11D0-B27A-
0004AC575688" height=0 width=0></OBJECT>

<SCRIPT language="VBScript">
Function passmac_OnScreenChanged (inState, inStart, inEnd)
Dim strText, i
strText = ""
For i = 1 To 24
strText = strText + passmac.GetString (i, 1, 80) + Chr(10)
Next
document.zform.txtscreen.Value = strText
End Function

Function passmac_OnHostKey (keyName)
document.zform.hostkey.value = keyName
End Function

Function passmac_OnLocalKey (keyName)
document.zform.localkey.value = keyName
End Function
```

```
</SCRIPT>

<BODY>
<FORM name=zform >
<INPUT type="button" name="btnConnect" value="Connect"
onclick="passmac.ConnectPS('A')">
Local Key: <INPUT type=text value="" name=localkey size=10 readonly>
Host Key: <INPUT type=text value="" name=hostkey size=10 readonly>
<TEXTAREA name=txtscreen rows=25 cols=81 wrap=virtual
readonly></TEXTAREA>
</FORM>
</BODY>

</HTML>
```

OnHostKey

Description

This is an event driven function. This function is called whenever the user presses a key that causes a data transmission to be sent to the host. This includes keys such as the ENTER or PF1 key.

Syntax

Sub **OnHostKey** (strKeyName)
End Sub

Parameters

strKeyName	String, input
	A text string that indicates the host key sent to the host.

Remarks

This routine is called before anything is actually sent to the host. When this routine returns then the data is actually sent to the host.

For the list of valid text strings that indicate the host key sent refer to the Macro Host Key Names for 3270, 5250 and VT section described in the SendHostKeys function. 3270, 5250 or VT Keys are enclosed within <...>. The following keys do not generate an event: ALTCURSOR, CURSOR RULER, CURSOR BLINK, Edit-xxxx keys, JUMP xxxx keys, and SELECT xxxx keys.

Zephyr does not recommend using event driven functions inside of a PASSPORT macro. Instead, these functions should be used in an external program or web page that uses the PASSMAC.DLL object to interface with the PASSPORT terminal emulator program. Below is an example of using this function inside of a web page using VBScript.

Example

```
<HTML>

<OBJECT name=passmac classid="CLSID:6440DEE3-7072-11D0-B27A-
0004AC575688" height=0 width=0></OBJECT>

<SCRIPT language="VBScript">
Function passmac_OnScreenChanged (inState, inStart, inEnd)
Dim strText, i
strText = ""
For i = 1 To 24
strText = strText + passmac.GetString (i, 1, 80) + Chr(10)
Next
document.zform.txtscreen.Value = strText
End Function

Function passmac_OnHostKey (keyName)
document.zform.hostkey.value = keyName
End Function

Function passmac_OnLocalKey (keyName)
document.zform.localkey.value = keyName
End Function
</SCRIPT>
```

```
<BODY>
<FORM name=zform >
<INPUT type="button" name="btnConnect" value="Connect"
onclick="passmac.ConnectPS('A')">
Local Key: <INPUT type=text value="" name=localkey size=10 readonly>
Host Key: <INPUT type=text value="" name=hostkey size=10 readonly>
<TEXTAREA name=txtscreen rows=25 cols=81 wrap=virtual
readonly></TEXTAREA>
</FORM>
</BODY>

</HTML>
```

OnLocalKey

Description

This is an event driven function. This function is called whenever the user presses any key that causes text to be entered into an input field, a local screen editing function, or a data transmission to be sent to the host.

Syntax

Sub **OnLocalKey** (strKeyName)
End Sub

Parameters

strKeyName	String, input
	A text string that indicates a text character entered, or the host key sent to the host.

Remarks

This routine is called before anything is actually sent to the host. When this routine returns then the data is actually displayed on the terminal emulator screen or sent to the host.

If the user presses a text character, just the text character is in the text string. For example if the user types the letter A, the text string will contain "A".

For the list of valid text strings that indicate the host key sent refer to the Macro Host Key Names for 3270, 5250 and VT section described in the SendHostKeys function. 3270, 5250 or VT Keys are enclosed within <...>. The following keys do not generate an event: ALTCURSOR, CURSOR RULER, CURSOR BLINK, Edit-xxxx keys, JUMP xxxx keys, and SELECT xxxx keys.

Zephyr does not recommend using event driven functions inside of a PASSPORT macro. Instead, these functions should be used in an external program or web page that uses the PASSMAC.DLL object to interface with the PASSPORT terminal emulator program. Below is an example of using this function inside of a web page using VBScript.

Example

```
<HTML>

<OBJECT name=passmac classid="CLSID:6440DEE3-7072-11D0-B27A-
0004AC575688" height=0 width=0></OBJECT>

<SCRIPT language="VBScript">
Function passmac_OnScreenChanged (inState, inStart, inEnd)
Dim strText, i
strText = ""
For i = 1 To 24
strText = strText + passmac.GetString (i, 1, 80) + Chr(10)
Next
document.zform.txtscreen.Value = strText
End Function

Function passmac_OnHostKey (keyName)
document.zform.hostkey.value = keyName
End Function

Function passmac_OnLocalKey (keyName)
document.zform.localkey.value = keyName
```

```
End Function
</SCRIPT>

<BODY>
<FORM name=zform >
<INPUT type="button" name="btnConnect" value="Connect"
onclick="passmac.ConnectPS('A')">
Local Key: <INPUT type=text value="" name=localkey size=10 readonly>
Host Key: <INPUT type=text value="" name=hostkey size=10 readonly>
<TEXTAREA name=txtscreen rows=25 cols=81 wrap=virtual
readonly></TEXTAREA>
</FORM>
</BODY>

</HTML>
```

Programming with HLLAPI

Programming with HLLAPI Overview

HLLAPI (High Level Language Application Programming Interface) is the industry standard method for a Windows PC application to interface with a host using a 3270, 5250, VT, SCO-ANSI or Wyse-60 Windows terminal emulation program. Typical HLLAPI applications can present users with a friendlier interface to host applications and can be used to automate routine functions such as file transfer.

Note: PASSPORT supports both the 16-bit and 32-bit WOSA Windows HLLAPI Specification Version 1.1 and is compatible with the HLLAPI support found in the IBM Personal Communications/3270 terminal emulation software.

If you want to use a commercially distributed application that interfaces with a 3270 emulator via HLLAPI, configure the application as you would for use with the IBM PC/3270 package and proceed as normal.

HLLAPI Configuration

HLLAPI is automatically available in every session of PASSPORT. By default, each PASSPORT session that is opened is assigned an alphabetic character, ranging sequentially from A through Z. This character is called the HLLAPI short name.

Windows HLLAPI defines a standard IBM HLLAPI style API for the 16-bit and 32-bit versions of Windows. This encompasses both familiar IBM HLLAPI style routines and a set of Windows-specific extensions. Windows 32-bit HLLAPI support is provided by the file **Passhll.dll**. Windows 16-bit HLLAPI support is provided by the **Pashll16.dll** file.

PASSPORT does not support DOS EHLLAPI applications running in a DOS window. If you have a DOS EHLLAPI application you will need to contact the supplier of the EHLLAPI program and obtain their 16-bit or 32-bit Windows version of the product.

If Your HLLAPI Application Fails

If you are having trouble with your HLLAPI application, please check the following:

- Make sure that the current path contains a path to the Passhll.dll (for 32-bit) or Pashll16.dll (for 16-bit). Also, the HLLAPI application must be configured to load the correct HLLAPI DLL, either Passhll.dll (for 32-bit) or Pashll16.dll (for 16-bit).
- Make sure that you have the correct Passhll.dll (for 32-bit) on your PC.
- Ensure the Passhll.dll file size and date should be approximately 96 kb and 3/16/99 or later respectively.
- If you previously copied the older PASSHLL.DLL to another directory other than the PASSPORT directory, you must copy the new PASSHLL.DLL to that directory as well.
- Make sure the HLLAPI application is either a 16-bit or 32-bit HLLAPI Windows application. DOS HLLAPI applications are not supported.
- Make sure that the session short name is correct. By default PASSPORT assigns sequentially A to session 1 up to Z for session 26. If your HLLAPI application is talking to E and you are running just one session configured as A, there will never be communication between the application and the emulator. Either reconfigure the PASSPORT session short name in the Session Profile – HLLAPI tab or reassign the session name in your application.
- Some HLLAPI applications require configuration for specific emulators. As mentioned earlier, you should configure the HLLAPI application for use with the IBM Personal Communications/3270 Software Version 3.0 and higher.

  For example, with some versions of SAS Connect

  OPTIONS SET = VQDLLNAME “PASSHLL.DLL”
- If you still cannot use the HLLAPI application with PASSPORT, disconnect your current session and configure PASSPORT to run a HLLAPI trace.

To run a HLLAPI trace:

WEB TO HOST

1. Log off your mainframe session.
2. Disconnect the session using the **Communication→Disconnect** menu command.
3. Select **Communication→Traces→HLLAPI Trace** menu command.
4. Reestablish the session using the **Communication→Connect** menu command.
5. Try the HLLAPI application again.
6. Once you reach the point of failure, exit the HLLAPI program and PASSPORT. The file Ehtrace.log, which will be located in the current working directory (C:\ by default), contains a trace of all the HLLAPI function calls along with parameters.
7. Copy the file to another directory or rename the file.

Warning: Your trace information will be overwritten if you restart the HLLAPI program!

PC TO HOST

1. Log off your mainframe session.
2. Disconnect the session using the **Communication→Disconnect** menu command.
3. Select the **Communication→Setup** menu command.
4. Select **Diagnostics** tab.
5. Enable the **HLLAPI Trace** option, then choose the **Connect** button to reestablish the connection.
6. Try the HLLAPI application again.
7. Once you reach the point of failure, exit the HLLAPI program and PASSPORT. The file **Ehtrace.log**, which will be located in the PASSPORT folder, contains a trace of all the HLLAPI function calls along with parameters.
8. Copy the file to another directory or rename the file.

Warning: Your trace information will be overwritten if you restart the HLLAPI program!

Note: Look at the Ehtrace.log file with a text editor, such as Notepad or Wordpad, and see if there is any HLLAPI communication in the file. If there is no HLLAPI communication in the trace file, you have probably not managed to get the application to talk to the emulator. If this is the case, go back and carefully repeat steps 1 through 5 and try again.

If your HLLAPI trace file does show signs of activity, send the file to Zephyr Technical Support. A 3270, 5250, VT or Wyse-60 data stream trace taken at the same time will also help.

To send your trace files to Zephyr Technical Support:

First, fill out a problem report at **http://www.zephyrcorp.com/support.htm** This ensures that we can track your problem and route it to the correct support technician. Next, email the trace file or files along with a detailed description of the problem to **support@zephyrcorp.com** or send the trace on disk to Zephyr Technical Support.

Supported HLLAPI Functions

The following HLLAPI functions are supported in this release of PASSPORT:

(1) Connect Presentation Space

(2) Disconnect Presentation Space

(3) Send Key

(4) Wait

(5) Copy Presentation Space

(6) Search Presentation Space

(7) Query Cursor Location

(8) Copy Presentation Space To String

(9) Set Session Parameters

(10) Query Sessions

(11) Reserve

(12) Release

(13) Copy OIA

(14) Query Field attribute

(15) Copy String To Presentation Space

(18) Pause

(20) Query System

(21) Reset System

(22) Query Session Status

(23) Start Host Notification

(24) Query Host Update

(25) Stop Host Notification

(30) Search Field

(31) Find Field Position

(32) Find Field Length

(33) Copy String To Field

(34) Copy Field To String

(40) Set Cursor

(50) Start Keystroke Intercept

(51) Get Key

(52) Post Intercept Status

(53) Stop Keystroke Intercept

(90) Send File

(91) Receive File

(99) Convert Position Row/Col

(101) Connect Window Services

(102) Disconnect Window Services

(104) Windows Status

(4112) RunProfile

(4113) RunConnect

(4114) CloseDisconnect

(4272) FTPSendFile

(4273) FTPReceiveFile

WinHLLAPICleanup ( )

WinHLLAPIStartup ( )

Unsupported HLLAPI Functions

The following HLLAPI functions are not supported in this release of PASSPORT:

(17) Storage Manager

(41) Start Close Intercept

(42) Query Close Intercept

(43) Stop Close Intercept

(60) Lock Presentation Space API

(61) Lock Window Services API

(103) Query Windows Coordinates

(105) Change Presentation Space Window Name

(120) Connect Structured Fields

(121) Disconnect Structured Fields

(122) Query Communications Buffer Size

(123) Allocate Communications Buffer

(124) Free Communications Buffer

(125) Get Request Completion

(126) Read structured Fields

(127) Write Structured Fields

WinAsyncHLLAPI ( )

WinHLLAPIIsBlocking ( )

WinAsyncHLLAPICancelRequest ( )

WinHLLAPICancelBlockingCall ( )

WinHLLAPISetBlockingHook ( )

WinHLLAPIUnhookBlockingHook ( )

Troubleshooting

Error Messages and Codes

PASSTCP - Connection Time Out Error
The HOST_ADDRESS has failed to respond within 30 seconds. Please check to make sure the IP Host Address is correct, reconfigure, and then try starting PASSPORT again.

PASSTCP - Windows Sockets Startup Error - Return Code = XXXXX

PASSTCP - Unable to resolve IP Host address HOST_ADDRESS - Return Code = XXXXX

PASSTCP - Socket Send Error - Return Code = XXXXX

PASSTCP - Socket Receive Error - Return Code = XXXXX

PASSTCP - Unable to Create Socket - Return Code = XXXXX

PASSTCP - Asynchronous Select Error - Return Code = XXXXX

PASSTCP - Socket Connect Error - Return Code = XXXXX

PASSTCP - Connection Error - Return Code = XXXXX
An error has occurred attempting to connect to Host IP Address nnn.nnn.nnn.nnn . If you are using Microsoft SNA Server, or another SNA gateway using TCP/IP, the TN3270 service must be installed and configured on that gateway.

PASSTCP - TN3270E Device Type Rejected - Reason Code = 0
The CONNECT method was configured to request a specific printer, but the device name 'DEVICE_NAME' requested is the partner to some terminal session. You must either reconfigure PASSPORT or reconfigure your TN3270E Server.

PASSTCP - TN3270E Device Type Rejected - Reason Code = 1
The requested device name 'DEVICE_NAME' is already being used by another Telnet session.

PASSTCP - TN3270E Device Type Rejected - Reason Code = 2
The ASSOCIATE method was configured to request a specific printer. Either the device type is not a printer, or the device name 'DEVICE_NAME' is not a terminal. You must either reconfigure PASSPORT or reconfigure your TN3270E Server.

PASSTCP - TN3270E Device Type Rejected - Reason Code = 3
The device name 'DEVICE_NAME' is not known to the TN3270E Server. You must either reconfigure PASSPORT or reconfigure your TN3270E Server.

PASSTCP - TN3270E Device Type Rejected - Reason Code = 4
The TN3270E Server does not support the requested device type DEVICE_TYPE. You must either reconfigure PASSPORT or reconfigure your TN3270E Server.

PASSTCP - TN3270E Device Type Rejected - Reason Code = 5
The configured device name DEVICE_NAME is incompatible with the configured device type DEVICE_TYPE (such as terminal/printer mismatch). You must either reconfigure PASSPORT or reconfigure your TN3270E Server.

PASSTCP - TN3270E Device Type Rejected - Reason Code = 6
A device type or device name processing error has occurred.

PASSTCP - TN3270E Device Type Rejected - Reason Code = 7
The TN3270E server is unable to satisfy the type of connection request sent by PASSPORT. A specific terminal or printer session was requested but the server does not have such a pool of device names defined to it, or the ASSOCIATE command was used but no partner printers are defined to the server. You must either reconfigure PASSPORT or reconfigure your TN3270E Server.

PASSTCP Windows Sockets Error Codes XXXXX

The return code XXXXX below is an error code returned by the Windows Sockets Winsock.dll. The explanation below gives the reason for the error return code.

WSAEINTR (10004)
Interrupted function call.
A blocking operation was interrupted by a call to WSACancelBlockingCall.

WSAEACCES (10013)
Permission denied.
An attempt was made to access a socket in a way forbidden by its access permissions. An example is using a broadcast address for sendto without broadcast permission being set using setsockopt(SO_BROADCAST).Another possible reason for the WSAEACCES error is that when the bind function is called (on Windows NT 4 SP4 or later), another application, service, or kernel mode driver is bound to the same address with exclusive access. Such exclusive access is a new feature of Windows NT 4 SP4 and later, and is implemented by using the SO_EXCLUSIVEADDRUSE option.

WSAEFAULT (10014)
Bad address.
The system detected an invalid pointer address in attempting to use a pointer argument of a call. This error occurs if an application passes an invalid pointer value, or if the length of the buffer is too small. For instance, if the length of an argument, which is a SOCKADDR structure, is smaller than the sizeof(SOCKADDR).

WSAEINVAL (10022)
Invalid argument.
Some invalid argument was supplied (for example, specifying an invalid level to the setsockopt function). In some instances, it also refers to the current state of the socket—for instance, calling accept on a socket that is not listening.

WSAEMFILE (10024)
Too many open files.
Too many open sockets. Each implementation may have a maximum number of socket handles available, either globally, per process, or per thread.

WSAEWOULDBLOCK (10035)
Resource temporarily unavailable.
This error is returned from operations on nonblocking sockets that cannot be completed immediately, for example recv when no data is queued to be read from the socket. It is a nonfatal error, and the operation should be retried later. It is normal for WSAEWOULDBLOCK to be reported as the result from calling connect on a nonblocking SOCK_STREAM socket, since some time must elapse for the connection to be established.

WSAEINPROGRESS (10036)
Operation now in progress.
A blocking operation is currently executing. Windows Sockets only allows a single blocking operation—per- task or thread—to be outstanding, and if any other function call is made (whether or not it references that or any other socket) the function fails with the WSAEINPROGRESS error.

WSAEALREADY (10037)
Operation already in progress.
An operation was attempted on a nonblocking socket with an operation already in progress—that is, calling connect a second time on a nonblocking socket that is already connecting, or canceling an asynchronous request (WSAAsyncGetXbyY) that has already been canceled or completed.

WSAENOTSOCK (10038)
Socket operation on nonsocket.
An operation was attempted on something that is not a socket. Either the socket handle parameter did not reference a valid socket, or for select, a member of an fd_set was not valid.

WSAEDESTADDRREQ (10039)
Destination address required.
A required address was omitted from an operation on a socket. For example, this error is returned if sendto is called with the remote address of ADDR_ANY.

WSAEMSGSIZE (10040)
Message too long.
A message sent on a datagram socket was larger than the internal message buffer or some other network limit, or the buffer used to receive a datagram was smaller than the datagram itself.

WSAEPROTOTYPE (10041)
Protocol wrong type for socket.
A protocol was specified in the socket function call that does not support the semantics of the socket type requested. For example, the ARPA Internet UDP protocol cannot be specified with a socket type of SOCK_STREAM.

WSAENOPROTOOPT (10042)
Bad protocol option.
An unknown, invalid or unsupported option or level was specified in a getsockopt or setsockopt call.

WSAEPROTONOSUPPORT (10043)
Protocol not supported.
The requested protocol has not been configured into the system, or no implementation for it exists. For example, a socket call requests a SOCK_DGRAM socket but specifies a stream protocol.

WSAESOCKTNOSUPPORT (10044)
Socket type not supported.
The support for the specified socket type does not exist in this address family. For example, the optional type SOCK_RAW might be selected in a socket call, and the implementation does not support SOCK_RAW sockets at all.

WSAEOPNOTSUPP (10045)
Operation not supported.
The attempted operation is not supported for the type of object referenced. Usually this occurs when a socket descriptor to a socket that cannot support this operation is trying to accept a connection on a datagram socket.

WSAEPFNOSUPPORT (10046)
Protocol family not supported.
The protocol family has not been configured into the system or no implementation for it exists. This message has a slightly different meaning from WSAEAFNOSUPPORT. However, it is interchangeable in most cases, and all Windows Sockets functions that return one of these messages also specify WSAEAFNOSUPPORT.

WSAEAFNOSUPPORT (10047)
Address family not supported by protocol family.
An address incompatible with the requested protocol was used. All sockets are created with an associated address family (that is, AF_INET for Internet Protocols) and a generic protocol type (that is, SOCK_STREAM). This error is returned if an incorrect protocol is explicitly requested in the socket call, or if an address of the wrong family is used for a socket, for example, in sendto.

WSAEADDRINUSE (10048)
Address already in use.
Typically, only one usage of each socket address (protocol/IP address/port) is permitted. This error occurs if an application attempts to bind a socket to an IP address/port that has already been used for an existing socket, or a socket that was not closed properly, or one that is still in the process of closing. For server applications that need to bind multiple sockets to the same port number, consider using setsockopt (SO_REUSEADDR). Client applications usually need not call bind at all—connect chooses an unused port automatically. When bind is called with a wildcard address (involving ADDR_ANY), a WSAEADDRINUSE error could be delayed until the specific address is committed. This could happen with a call to another function later, including connect, listen, WSAConnect, or WSAJoinLeaf.

WSAEADDRNOTAVAIL (10049)
Cannot assign requested address.
The requested address is not valid in its context. This normally results from an attempt to bind to an address that is not valid for the local machine. This can also result from connect, sendto, WSAConnect, WSAJoinLeaf, or WSASendTo when the remote address or port is not valid for a remote machine (for example, address or port 0).

WSAENETDOWN (10050)
Network is down.
A socket operation encountered a dead network. This could indicate a serious failure of the network system (that is, the protocol stack that the Windows Sockets DLL runs over), the network interface, or the local network itself.

WSAENETUNREACH (10051)
Network is unreachable.
A socket operation was attempted to an unreachable network. This usually means the local software knows no route to reach the remote host.

WSAENETRESET (10052)
Network dropped connection on reset.
The connection has been broken due to keep-alive activity detecting a failure while the operation was in progress. It can also be returned by setsockopt if an attempt is made to set SO_KEEPALIVE on a connection that has already failed.

WSAECONNABORTED (10053)
Software caused connection abort.
An established connection was aborted by the software in your host machine, possibly due to a data transmission time-out or protocol error.

WSAECONNRESET (10054)
Connection reset by peer.
An existing connection was forcibly closed by the remote host. This normally results if the peer application on the remote host is suddenly stopped, the host is rebooted, or the remote host uses a hard close (see setsockopt for more information on the SO_LINGER option on the remote socket.) This error may also result if a connection was broken due to keep-alive activity detecting a failure while one or more operations are in progress. Operations that were in progress fail with WSAENETRESET. Subsequent operations fail with WSAECONNRESET.

WSAENOBUFS (10055)
No buffer space available.
An operation on a socket could not be performed because the system lacked sufficient buffer space or because a queue was full.

WSAEISCONN (10056)
Socket is already connected.
A connect request was made on an already-connected socket. Some implementations also return this error if sendto is called on a connected SOCK_DGRAM socket (for SOCK_STREAM sockets, the to parameter in sendto is ignored) although other implementations treat this as a legal occurrence.

WSAENOTCONN (10057)
Socket is not connected.
A request to send or receive data was disallowed because the socket is not connected and (when sending on a datagram socket using sendto) no address was supplied. Any other type of operation might also return this error—for example, setsockopt setting SO_KEEPALIVE if the connection has been reset.

WSAESHUTDOWN (10058)
Cannot send after socket shutdown.
A request to send or receive data was disallowed because the socket had already been shut down in that direction with a previous shutdown call. By calling shutdown a partial close of a socket is requested, which is a signal that sending or receiving, or both have been discontinued.

WSAETIMEDOUT (10060)
Connection timed out.
A connection attempt failed because the connected party did not properly respond after a period of time, or the established connection failed because the connected host has failed to respond.

WSAECONNREFUSED (10061)
Connection refused.
No connection could be made because the target machine actively refused it. This usually results from trying to connect to a service that is inactive on the foreign host—that is, one with no server application running.

WSAEHOSTDOWN (10064)
Host is down.
A socket operation failed because the destination host is down. A socket operation encountered a dead host. Networking activity on the local host has not been initiated. These conditions are more likely to be indicated by the error WSAETIMEDOUT.

WSAEHOSTUNREACH (10065)
No route to host.
A socket operation was attempted to an unreachable host. See WSAENETUNREACH.

WSAEPROCLIM (10067)
Too many processes.
A Windows Sockets implementation may have a limit on the number of applications that can use it simultaneously. WSAStartup may fail with this error if the limit has been reached.

WSASYSNOTREADY (10091)
Network subsystem is unavailable.
This error is returned by WSAStartup if the Windows Sockets implementation cannot function at this time because the underlying system it uses to provide network services is currently unavailable. Users should check:

- That the appropriate Windows Sockets DLL file is in the current path.
- That they are not trying to use more than one Windows Sockets implementation simultaneously. If there is more than one Winsock DLL on your system, be sure the first one in the path is appropriate for the network subsystem currently loaded.
- The Windows Sockets implementation documentation to be sure all necessary components are currently installed and configured correctly.

WSAVERNOTSUPPORTED (10092)
Winsock.dll version out of range.
The current Windows Sockets implementation does not support the Windows Sockets specification version requested by the application. Check that no old Windows Sockets DLL files are being accessed.

WSANOTINITIALISED (10093)
Successful WSAStartup not yet performed.
Either the application has not called WSAStartup or WSAStartup failed. The application may be accessing a socket that the current active task does not own (that is, trying to share a socket between tasks), or WSACleanup has been called too many times.

WSAEDISCON (10101)
Graceful shutdown in progress.
Returned by WSARecv and WSARecvFrom to indicate that the remote party has initiated a graceful shutdown sequence.

WSATYPE_NOT_FOUND (10109)
Class type not found.
The specified class was not found.

WSAHOST_NOT_FOUND (11001)
Host not found.
No such host is known. The name is not an official host name or alias, or it cannot be found in the database(s) being queried. This error may also be returned for protocol and service queries, and means that the specified name could not be found in the relevant database.

WSATRY_AGAIN (11002)
Nonauthoritative host not found.
This is usually a temporary error during host name resolution and means that the local server did not receive a response from an authoritative server. A retry at some time later may be successful.

WSANO_RECOVERY (11003)
This is a nonrecoverable error.
This indicates some sort of nonrecoverable error occurred during a database lookup. This may be because the database files (for example, BSD-compatible HOSTS, SERVICES, or PROTOCOLS files) could not be found, or a DNS request was returned by the server with a severe error.

WSANO_DATA (11004)
Valid name, no data record of requested type.
The requested name is valid and was found in the database, but it does not have the correct associated data being resolved for. The usual example for this is a host name-to-address translation attempt (using gethostbyname or WSAAsyncGetHostByName) which uses the DNS (Domain Name Server). An MX record is returned but no A record—indicating the host itself exists, but is not directly reachable.

WSA_INVALID_HANDLE (OS dependent)
Specified event object handle is invalid.
An application attempts to use an event object, but the specified handle is not valid.

WSA_INVALID_PARAMETER (OS dependent)
One or more parameters are invalid.
An application used a Windows Sockets function which directly maps to a Win32 function. The Win32 function is indicating a problem with one or more parameters.

WSA_IO_INCOMPLETE (OS dependent)
Overlapped I/O event object not in signaled state.
The application has tried to determine the status of an overlapped operation which is not yet completed. Applications that use WSAGetOverlappedResult (with the fWait flag set to FALSE) in a polling mode to determine when an overlapped operation has completed, get this error code until the operation is complete.

WSA_IO_PENDING (OS dependent)
Overlapped operations will complete later.
The application has initiated an overlapped operation that cannot be completed immediately. A completion indication will be given later when the operation has been completed.

WSA_NOT_ENOUGH_MEMORY (OS dependent)
Insufficient memory available.
An application used a Windows Sockets function that directly maps to a Win32 function. The Win32 function is indicating a lack of required memory resources.

WSA_OPERATION_ABORTED (OS dependent)
Overlapped operation aborted.
An overlapped operation was canceled due to the closure of the socket, or the execution of the SIO_FLUSH command in WSAIoctl.

WSAINVALIDPROCTABLE (OS dependent)
Invalid procedure table from service provider.
A service provider returned a bogus procedure table to Ws2_32.dll. (Usually caused by one or more of the function pointers being null.)

WSAINVALIDPROVIDER (OS dependent)
Invalid service provider version number.
A service provider returned a version number other than 2.0.

WSAPROVIDERFAILEDINIT (OS dependent)
Unable to initialize a service provider.
Either a service provider's DLL could not be loaded (LoadLibrary failed) or the provider's WSPStartup/NSPStartup function failed.

WSASYSCALLFAILURE (OS dependent)
System call failure.
Returned when a system call that should never fail does. For example, if a call to WaitForMultipleObjects fails or one of the registry functions fails trying to manipulate the protocol/name space catalogs.

Program Check Errors

PROG 750
An Invalid 3270 command was received.

PROG 751
A START FIELD EXTENDED, MODIFY FIELD, or SET ATTRIBUTE order that specified an invalid character set was received.

PROG 752
A SET BUFFER ADDRESS, REPEAT TO ADDRESS, or ERASE UNPROTECTED TO ADDRESS order that specified an invalid address was received.

PROG 753
A READ MODIFIED, READ MODIFIED ALL, or READ BUFFER command that also contained data was received.

PROG 754
One of the following commands was received without required parameters:

- SET BUFFER ADDRESS
- REPEAT TO ADDRESS
- ERASE UNPROTECTED TO ADDRESS
- START FIELD
- START FIELD EXTENDED
- MODIFY FIELD
- SET ATTRIBUTE
- GRAPHIC ESCAPE

PROG 755
Invalid character code was received.

PROG 756
A WRITE STRUCTURED FIELD command was received with an invalid structured field.

PROG 758
A SET REPLY MODE command was received with an invalid mode.

PROG 759
A WRITE STRUCTURED FIELD command was received with an invalid structured field length.

Running a Trace

The PASSPORT trace feature is designed to assist in the diagnosis of problems that can occur during communications and terminal emulation. Information captured in a trace file can be analyzed by the Zephyr technical support and programming staff to assist with problem resolution. Your site administrator may ask you to assist in running diagnostic traces. After running the traces, you can email them to your site administrator.

PASSPORT PC TO HOST supports three types of tracing:

- **Data Stream Trace** - This type of trace is helpful in resolving problems involving the SNA BIND attributes, file transfer, host printing, HLLAPI and other problems involving the TN3270, TN5250, VT, SCO-ANSI or Wyse data stream.
- **Low Level Trace** - This type of trace is helpful in resolving problems involving the establishment of a connection or problems that deal with the actual communication device driver.
- **HLLAPI Trace** - This type of trace is helpful in resolving problems with HLLAPI applications.

To run one or more traces:

1. Log off your mainframe session.
2. Close the session using the **Communication→Disconnect** menu command.
3. Enable the traces you want to run in the **Diagnostics** tab of the **Communication→Setup** dialog box.
4. Reestablish the session using the **Communication→Connect** menu command.
5. Log on to your mainframe and perform your terminal emulation activities.
6. Once you reach the point of failure, log off and exit PASSPORT.
7. Copy the trace files to another folder or rename them.

Note: Your trace files will be overwritten if you restart PASSPORT!

Your trace files will be in the following folders:

- The Data Stream Trace file is called Trace_xx.log (where xx is equal to the ASCII value of the HLLAPI short name for the session) and is located in the PASSPORT working directory.
- The Low Level Trace file is called Tracell_xx.log (where xx is equal to the ASCII value of the HLLAPI short name for the session) and is located in the PASSPORT working directory.
- The HLLAPI Trace file is called Ehtrace.log and is located in the PASSPORT working directory.

To view the trace files:

- All of the trace files are ASCII text data and may be viewed with Notepad, Wordpad or some other text-based editor.

To send your trace files to Zephyr Technical Support

- First, fill out a problem report at **http://www.zephyrcorp.com/techsupport/ask.html**. This ensures that we can track your problem and route it to the correct support technician. Next, mail the trace file or files along with a detailed description of the problem to **support@zephyrcorp.com** or send the trace on disk to Zephyr Technical Support.

PASSPORT WEB TO HOST supports three types of tracing:

- **Data Stream Trace** - This type of trace is helpful in resolving problems involving the SNA BIND attributes, file transfer, host printing, and other problems involving the 3270, 5250, VT52, VT100, VT220, SCO-ANSI or Wyse data stream.
- **Low Level Trace** - This type of trace is helpful in resolving problems involving the establishment of a connection or problems that deal with the actual communication device driver. The data traced at this level is at the WinSock layer.
- **HLLAPI Trace** - This type of trace is helpful in resolving problems with HLLAPI applications.

To run a trace do the following:

1. Log off your mainframe session.
2. Choose **Communication→Disconnect** to terminate the session. (Or select the Disconnect button from the toolbar).
3. Choose **Communication→Traces→Data Stream Trace, Communication→Traces→Low Level Trace** or **Communication→Traces→HLLAPI Trace** to enable the approprite tracing facility. A check mark will appear next to each menu item when activated.
4. Choose **Communication→Connect** to reestablish the session. (Or select the **Connect** button from the toolbar).
5. Log on to your host and perform your terminal emulation activities.
6. Once you reach the point of failure, then log off your host session.
7. Choose **Communication→Disconnect** to terminate the session. (Or select the **Disconnect** button from the toolbar).
8. Choose **Communication→Traces→Data Stream Trace, Communication→Traces→Low Level Trace** or **Communication→Traces→HLLAPI Trace** to disable tracing for each type of tracing that is enabled.
9. Copy the trace files to another folder or rename them. There will be two trace files in the "\Program Files\eClient Local Files\" folder of the drive where the Windows operating system is installed with the following names:
   - **Trace_xx.log**
   - **TraceII_xx.log**

     (where xx is equal to the ASCII value of the HLLAPI short name for the session, i.e. for session A xx is 41, for session B xx is 42, etc.).

The HLLAPI Trace file will be located in the current working directory on the workstation where the traces are run (C:\ by default) and will have the following name:

- **Ehtrace.log**

Note: Your trace files will be overwritten if you use **Communication→Connect** to reestablish the session prior to disabling the traces (step 8).

Note: All of the trace files are ANSI text data and may be viewed with Notepad or any text-based editor.

10. Send these trace files to your site administrator.

Important Note: If you do not disable traces after running your trace and you connect your session again, your previous trace files will be overwritten and all data contained in those previous trace files will be lost.

Miscellaneous

Attachmate API Compatibility

PASSPORT provides an API (application programming interface) compatible with the object oriented Attachmate EXTRA!® OLE automation support API. This interface exposes terminal emulation objects such as the entire group of sessions, a single session, screen, keypad, etc.

This API can be used by C++, Visual Basic, VBScript, JavaScript and other applications that need to access host data and/or control multiple host sessions. This API is also used when running Attachmate EXTRA!® macros that have been converted to run as PASSPORT macros.

The PASSOBJ.DLL file provides this API interface. This DLL is registered with Windows when the PASSPORT ActiveX component is downloaded.

There is one change that needs to be made to your API application. The parameter for the CreateObject statement must be changed from "EXTRA.System" to "PASSPORT.System". This statement is usually the first statement that references the API.

An example for Attachmate EXTRA!® :

SET oSystem = CreateObject("EXTRA.System")

An example for PASSPORT:

SET oSystem = CreateObject("PASSPORT.System")

Attachmate Macro Support

Zephyr has developed a software program that converts Attachmate EXTRA!® macros from the proprietary Attachmate format into standard Visual Basic for use with PASSPORT. The macro conversion is provided as a free support service for Zephyr customers under a current Priority Maintenance Plan or with a current annual Subscription License for the PASSPORT software.

The Zephyr tool can be of great value for Attachmate customers who want to save money on host access using the Zephyr PASSPORT terminal emulation, but has a sizable number of EXTRA! macros that make it difficult to transition to another host access solution.

Customers or prospects who want to have Attachmate macros converted for production use, or as a test, should email them to support@zephyrcorp.com.

SLP Load Balancing

Load balancing is an extended access feature of TN3270E and TN5250E. To utilize TN3270E and TN5250E extended features, both server and client must support and be configured to participate.

Load balancing is a Communications Server function that uses **Service Location Protocol (SLP)** to dynamically balance LU (host-to-workstation) sessions by distributing them to the communications server with the smallest load. Clients using TCP/IP protocol and that support load balancing have the ability to query participating TN3270E and TN5250E servers and connect to the least loaded host gateway using SLP.

Service Location Protocol (SLP), a new Internet Engineering Task Force (IETF) standards-track protocol, provides a method of dynamic allocation of service requests among a set of servers with a level of workload balancing.

SLP is designed to simplify the discovery and use of network resources, including TN3270E and TN5250E servers, Web servers, fax machines, printers, file systems, databases and future services. SLP is a specification that defines a generic method to perform the following tasks:

- Service Registration
- Service Advertisement
- Service Query

Specialized components called Agents perform tasks and support services:

User Agent	Support for Service Query functions Acquire/request service information for user applications
Service Agent	Service Registration and Service Advertisement
Directory Agent	Collects service information from Service Agents which is later requested by User Agents in Intranet

Services are described by configuring attributes associated with a type of service. A User Agent can select the appropriate service by specifying the attribute that it needs in a Service Request. When a Service Request is returned, it contains a URL (Uniform Resource Locator) pointing to the Service desired, and other information needed by the User Agent.

SLP can reduce overall network traffic by using a Scope parameter to manage client service requests. Scope is essentially a grouping method to organize servers into named groups. Scope values are defined by the network administrator, and may represent departments, regions or organizations. Different scopes can be assigned for different services provided on the server if desired. Clients participating in SLP Load Balancing may submit both scoped and unscoped service requests.

Secure Sockets Layer (SSL)

The **Secure Sockets Layer (SSL)** protocol was developed by Netscape Communications Corporation to provide security and privacy over the Internet. Due to the widespread use of SSL, it can already be considered a de facto standard. PASSPORT uses server authentication to make sure that you are authorized to connect to a specific Telnet communication server. In this phase the server sends a digital certificate to PASSPORT. After authentication is validated using a combination of public and private keys, subsequent data transmissions are encrypted using one of the following supported cryptographic algorithms: RC2, RC4, RC5, DES, or Triple-DES. The public key certificates follow the X.509 syntax.

Non-secure Telnet sessions typically use TCP Port 23. **Secure Telnet** sessions using SSL typically use TCP Port 992, but may be re-configured to any available TCP Port desired.