Tom Brennan Software

Vista tn3270 Emulator

Vista tn3270 Emulator

Contents

Overview

Vista tn3270 is a Microsoft Windows application designed to emulate IBM standard 3270 series terminals in a TN3270 (TCPIP) environment provided by a standard winsock interface. Vista tn3270 version 1.20 (and above) runs on Windows 95, 98, NT, and 2000.

Vista is designed with one specific purpose in mind: To provide a terminal emulation connection to IBM mainframe computers running such programs as TSO, CICS, CMS, etc. See the Restrictions section in this document if you are not sure Vista is right for your environment.

If this program ***is*** what you need, then you may find Vista can do the job as well as many commercially available TN3270 emulators, and may even do some things they haven't implemented yet.

Defaults

It can be frustrating at first to try out a new terminal emulator simply because you don't know where all the keys are and what all the functions do. If you don't plan to immediately explore and alter some of the Vista defaults listed below, you may save yourself some trouble by at least taking a look at the settings as supplied:

 **Keyboard Defaults**

 **Toolbar Defaults**

 **Keypad Defaults**

Features

Vista has the following features and more. Certain items that you may not find on other emulators are highlighted in *green italics* below:

TN3270 SNA Terminal Support

With Extended Attributes, Highlighting, Reverse Video, Graphics-Escape character set, etc. Emulates most functions of 327x terminal models 2, 3, 4, 5, along with user specified screen sizes up to 72 lines by 200 characters wide.

Easy Connect/Reconnect

Quick connection to various hosts, without needing a separate configuration program
Easy reconnect, and easy terminal model alteration
Up to 26 multiple host sessions (labelled A through Z)
Assign an Alias to a host name or IP address

Rich set of Display Options and Special Functions

Large set of bitmap fonts for easy reading at most screen resolutions
Window auto-sized to fonts
Fully customizable screen colors *with brightness control*
Multiple Cut/Paste buffers (up to 9)
Rich set of Copy and Paste methods such as *PasteWindow text stream formatting*
Smart selection of words on screen using SelectJCL function
Smart replace of words on screen using PasteJCL function
Screen level Undo/Redo
Smart Row/Column indicator shows actual data column when in the ISPF editor
Option to prefix all input with spaces, to avoid 3270 null character problems
Completely definable window title and status bar text, with variables
Type ahead buffer
Auto keyboard unlock
Capture screen to a file
Screen print *with definable settings for each terminal type*
Disable Auto-Skip and/or Numeric Lock
Destructive or non-destructive BackSpace
Blinking text shows as blinking, italic, or normal text
Full control over Cursor size/shape and blink/noblink
Locator cursor action to help you find it on complex screens
Horizontal, Vertical, or Crosshairs ruler of any color follows cursor
ToolHelp and other Win95 lookalike options, even when using Win3.1
Support for McGill's PCPRINT TSO command

Screen "Hot Spot" support
Each session can have a different color icon and screen border, to make window switching easier

Fully Customizable Keyboard

Any key on a standard 101 keyboard can be modified
Each key can be set to repeat or not when held down
Each key can be set to be buffered or set to interrupt a locked screen
A Find function is available in the keyboard editor to find key functions
Mouse keys can be set to issue any function or macro
Shift, Ctrl, and Alt keys can be set to functions, while retaining their normal use
CapsLock and Numlock can be redefined, or even disabled
Special non-3270 functions are available such as BottomHome and BackNewLine

Customizable Toolbar and Popup Keypads

Toolbar buttons can be added, moved, or deleted as needed
Any toolbar button can be assigned any key or macro function
Shift, Ctrl, and Alt keys can modify a toolbar button's function

File Transfer Between PC and Host

IND$FILE transfer is supported for TSO and CMS in either WSF (Write Structured Field) or CUT (Control Unit Terminal) mode.

Macro/Script processing language

Automatically record macro scripts, or create them yourself in any text editor
Assign macros to the keyboard, mouse, or toolbar and keypad buttons
Macro screen input is typed as soon as screen unlocks, with no delay timer needed
Initiate file transfers from macros

International Support

Code pages supplied for various countries
Keyboard layout options with escape key functions for accented characters

Thin Code that runs on all 32bit Window's platforms

Vista provides all these features and more in an very fast-loading and fast-operating emulation package. A little over 2M of disk storage for the average installation. Gets you connected quickly even on older PC's.

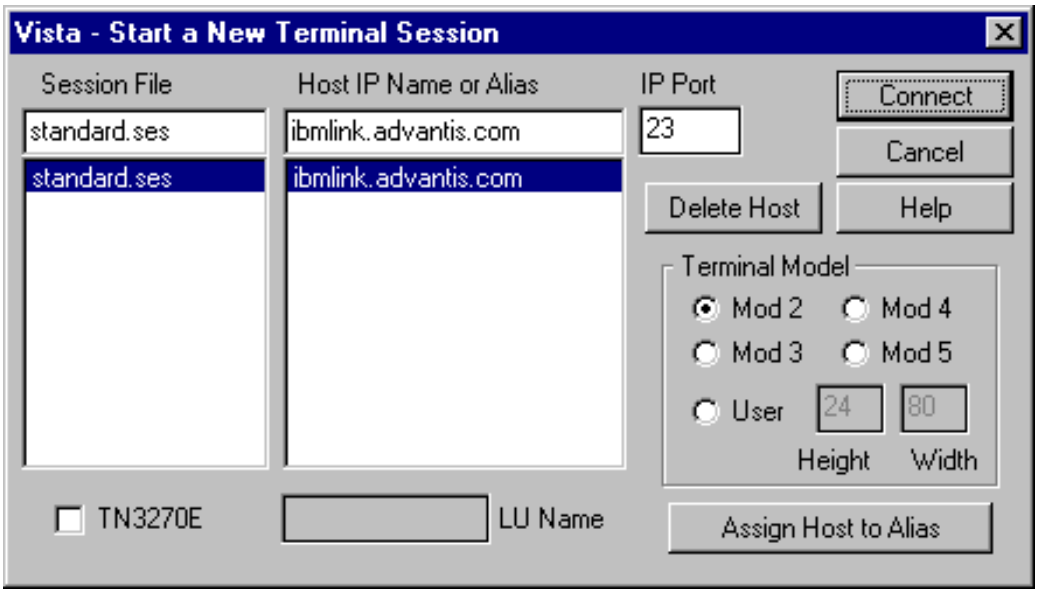
And for Windows 3.1 or newer, a 16bit version is available with most of the features listed above.

Connecting

There are 2 methods to start Vista: Manual and Automatic. After installation you should see two Vista icons. The one marked "Vista" is for Manual startup. The one marked "Vista Standard Session" is automatic.

Manual Startup

When Vista starts with no command line parameter, it will show the **Start a New Terminal Session** window, which looks like this:



To start a session, you will need to select (or type in) the **Host IP Name** of the TN3270 session you wish to establish, and optionally select a **Terminal Model** for the session. You can also select a pre-established **Session File**, which contains a saved set of Vista options, or just let it default to standard.ses.

Then press the **Connect** button to attempt to establish a TN3270 session.

Session window options are explained on another page

Automatic Startup

Starting a Vista session with a command line parameter pointing to a session file will bypass the Session window and attempt to establish a connection automatically. For example, setting the target command to:

```
C:\VISTA\VISTA16.EXE STANDARD.SES
```

...will startup Vista with the standard session file and attempt to connect to the host specified in that file.

Other command line parameters and INI variables are available if needed.in special cases

Additionally, Vista may be started by clicking on a hypertext link in a web browser that specifies a target URL of **TN3270://***hostname.* Before this will work, however, you must tell Windows to use Vista for that particular process. Here are some instructions for Windows 95, which I believe should also work with other versions of Windows:

1) Click **Start**, **Settings**, and **Control Panel**
2) Click the **View/Options** menu item, and press the **File Types** tab.
3) Find **URL:TN3270 Protocol** in the resulting list, and double click to **edit** that item.
4) In the resulting **actions** window, double click open to **edit** that action.
5) In the text box titled **Application used to perform action**, specify the drive, directory, and program name where you have Vista installed. By default this should be:

C:\VISTA32\VISTA.EXE

Close all the windows you just opened, and you should now be able to double click a TN3270 web-page URL and cause a new Vista session to open and automatically attempt to connect to the hostname specified in the URL.

Creating New Session Files

Vista comes with one session file named STANDARD.SES, and this one is used as a basis for any others you might want to create. To create a new session file, you can use the **File/Save Session As...** menu bar function, or you can also simply type in a new session name when you see the session selection window titled **Vista - Start a New Terminal Session**. This window comes up, for example, when a ReconnectAsk or NewSessionAsk function is done.

Both methods also have an option to create a new Vista icon in the Vista TN3270 folder.

You can also simply copy one session file to a new name using any Window's file-copy method.

A host name can be a name resolved by a Domain Name Server (DNS) such as tn3270.company.com, or a dot address such as 206.85.100.23

Vista supports 5 standard IBM terminal models:

Type	Rows x Columns
Model 2	24 x 80
Model 3	32 x 80
Model 4	43 x 80
Model 5	27 x 132
User	variable, up to 72 x 200

A session file contains most of the parameters (screen size, font, colors, etc.) for a particular Vista session. The default standard.ses is setup at installation time, but you can create your own by typing a new Session File name in the startup window.

Multiple Vista windows can use the same or different session files, as specified at startup or by command line parameter.

Command Line Options

Right click any Vista icon and select Properties, then click the Shortcut tab. The "Target" specification probably looks something like this:

```
c:\vista32\vista32.exe standard.ses
```

Items following the vista32.exe program name are Command Line Options. These can influence the operation of Vista, typically at startup.

Currently, these options are available:

sessionfilename *(or)* **/s** *sessionfilename*	Specifies the session file (*.ses) to be used at startup. If this parm is ommitted, Vista will not attempt to automatically connect to the host, and instead display the Connection window. Example: `c:\vista32\vista32.exe` **`mysession`**
/m *macrofilename*	Indicates the name of a macro to be executed at session startup. Note, this macro will only be executed on the first connection. If you want a macro to be executed at every connection, please use the Startup Macro specified on the General Options window. Example: `c:\vista32\vista32.exe` **`/m mymacro`**
/h *hostname*	Specifies the host name or ip address that should be used by this session at startup. A connection is attempted immediately, without any user interaction. This is considered a temporary hostname, and is not added to the hostname list. Example: `c:\vista32\vista32.exe` **`/h host.my.com`**
TN3270://*hostname*	This option results in the same action as the **/h** option specified above, but allows automatic startup and connection of an emulator by clicking on a hypertext link in a typical web browser. Typically, you will need to set your Windows default for the URL:TN3270 protocol so it points to the Vista emulator. See Connecting for more information on this.

INI Variables

Most of Vista's settings are stored in a session file (typically standard.ses), and in other files such as the keyboard and toolbar files. However, a few are stored in an INI file so they will be common to all sessions. The file name is **VISTA.INI** and is usually in your windows default directory (usually c:\windows or c:\winnt). ***Note:*** If **VISTA.INI** is found in the same directory as VISTA32.EXE, then that one will be used.

INI variables in the list below are created by Vista. You won't normally need to change these here, since they are updated automatically at each connection attempt:

LastSession	The last session file used
HostNames	A list of host names (really Alias names) that will appear in the Connection window. Don't change the order of this list, since it is related to other INI variable lists.
PortNums	A list of port numbers associated with each HostName
IPNames	IP addresses or names associated with each HostName. These may be different than the associated HostName, in the case of aliases.
TN3270E	A 1 or a 0 indicating if TN3270E protocol should be attempted for the associated host name
LUNames	LUNAMES associated with each HostName

The following INI variables are ***not*** set by Vista, but can be used in special cases if required. You'll need to type them in yourself and restart Vista to activate.

CopyBufferSize	The default copybuffersize is 64K (the maximum allowed). If you are having clipboard problems, you might try lowering this value a bit, such as the following example for 50K: Example: CopyBufferSize=50
RecordWait	When automatically recording a macro, Vista will place Wait statements with a default value of 20 after each action key statement. This gives the host 20 seconds to respond before the macro fails. If you need to increase this time, you can either edit the resulting macro, or add this INI variable prior to recording: Example: RecordWait=30
PatBltCursor	Vista's cursor is normally drawn on the screen using the "PatBltCursor" function of Windows. In the past, Vista used the “Rectangle” function which seemed to leave traces using some video boards. If you have cursor trouble, try setting PatBltCursor off by coding the following INI line: Example: PatBltCursor=0

HostnameFocus

The Connect Window normally moves the cursor (the "focus") to the Session File selection box when starting up or reconnecting. By using the HostnameFocus INI file option, you can cause the cursor to move to the Host Name box instead.

Example: HostnameFocus=1

NoResetClock

The RESET function normally clears the terminal of any locked status. With this option set though, reset only clears the X SYSTEM status, and will not clear the "clock" locked status (the picture of a little clock).

Example: NoResetClock=1

NoMacroOnReconnect

This option disables the Startup Macro when auto-reconnecting after a disconnect. Startup Macro will still be executed at the initial connection, and after any reconnections caused by functions like Reconnect or ReconnectAsk .

Example: NoMacroOnReconnect=1

TermTypeUser

By default, Vista passes terminal type IBM-3278-2-E to the telnet server when connected in “user” mode (user defined screen size). You can change this to mod5 (for example) if necessary by coding the following in the INI file:

Example: TermTypeUser=5

BeepPitchLow
BeepDurationLow
BeepPitchMedium
BeepDurationMedium
BeepPitchHigh
BeepDurationHigh

These INI options set the pitch and duration for the various PC speaker beeps (low, meduim, and high) which can be set for PC’s without sound cards using the Set Sounds button in the Display options window.

Pitch is specified in Hz, Duration in milliseconds.

Examples are shown below with their defaults:

BeepPitchLow=500
BeepDurationLow=300
BeepPitchMedium=1000
BeepDurationMedium=300
BeepPitchHigh=1500
BeepDurationHigh=300

FontAlreadyLoaded

Vista fonts are loaded at startup, and removed from memory when the last Vista window is closed. However, in some cases you may need to implement the fonts in the Windows registry permanently. If so, you will need to tell Vista the fonts are already loaded by:

Example: FontAlreadyLoaded=1

KeepAliveTime

Tells Vista to send a null signal to the TN3270 server periodically to keep a session active. A telnet IAC NOP is sent (x'FFF1'). Value is in seconds, and zero (the default) tells Vista not to send any keepalive. For example, if you want to send a keepalive every 60 seconds, code:

Example: KeepAliveTime=60

CloseOnDisconnect

Causes the Vista main window to close when a disconnect from the host occurs. Useful for hosts that issue a disconnect when you LOGOFF for example.

Example: CloseOnDisconnect=1

ShutDownMacro

Specifies a macro that will be run when the Vista main window is closed.

Example: ShutDownMacro=shutdown.mac

Here's an example of a macro which will check if ISPF edit is active at shutdown, and press PF3 a few times for you before closing the Vista session:

```
*********************************
* Shutdown macro
*********************************

If Screen(3,2,4) = "EDIT"
  If Screen(4,2,12) = "Command ===>"
    Key("PF3")
    Wait(10,status="unlocked")
    Key("PF3")
    Wait(10,status="unlocked")
    Key("PF3")
    Wait(10,status="unlocked")
    Key("PF3")
    Wait(10,status="unlocked")
  Endif
Endif
Key("ExitEmulator")

*********************************
```

Note that the ExitEmulator function should be specified to actually cause the emulator window to close.

Start a New Terminal Session

The following parameters are available when manually starting a new session:

Session File

Select an existing session file, or type a new name. A session file contains most of the parameters for a session such as font size, screen colors, and other options. Multiple Vista windows can share the same session file, but remember that each window stores parameters at close time. This means parameters will be set to those of the *last* window closed.

Host IP Name or Alias

This name is usually a Domain Name Server (DNS) name such as tn3270.company.com, or a dot address such as 206.85.100.23. It can also be an Alias name that points to either a DNS or dot address (see Assign Host to Alias below).

You will need to know this IP information before a connection can be established. Ask the network support people at your company if you aren't already connected. Also, you may need to ask them how to setup a TCP/IP connection on your PC.

IP names and their associated port numbers are stored in the VISTA.INI file in the windows directory, and not in the session file, so that they can be shared among sessions.

IP Port

Each **Host IP Name** has an associated **IP Port** number, which you can change in this field. Normally, TN3270 is defined to port 23, but in some cases your connection may need to use a different port number.

Delete Host

This button can be used to delete the selected **Host IP Name** if you want to clean up the list a bit.

Terminal Model

Vista can emulate 5 standard IBM 327x models:

Mod 2	24 lines by 80 columns
Mod 3	32 lines by 80 columns
Mod 4	43 lines by 80 columns
Mod 5	27 lines by 132 columns
User	Variable up to 72 x 200

Because of Host restrictions, a session needs to be stopped and restarted in order to change the **Terminal Model**

For each terminal model other than Mod 2, you may need to specify a LOGMODE parameter when logging on to the host. For example, to logon with a User specified size such as 56 x 127 (nice for viewing syslog), you might try something like:

```
LOGON APPLID=TSO,LOGMODE=D4A32XX3
```

Ask your local VTAM System Programmer to be sure.

Assign Host to Alias

Dot addresses and DNS IP names can be a bit cryptic to look at. Instead, you can type a more

descriptive name, such as "Main Computer" in the Host IP Name or Alias field, and then press the Assign Host to Alias button to relate the alias name to a real DNS name or dot address.

TN3270E

Set Vista to use TN3270E protocol instead of the default TN3270 standard mode. See Notes about TN3270E for more information.

LU Name

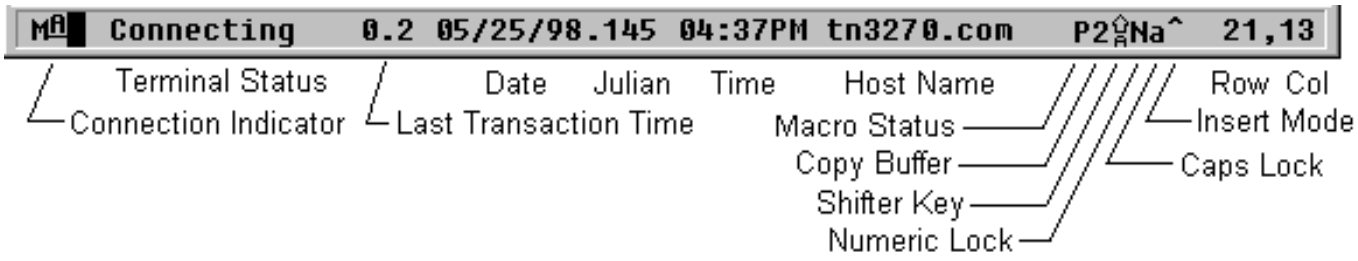
When in TN3270E mode, you have the option of specifying a VTAM 8 character LUNAME to be used when establishing the connection.

Connect Button

When this is pressed, Vista attempts to connect to the specified **Host IP Name.** If there is a problem, you may see an error window, such as:

Status Bar

The Status Bar is the text at the bottom of the Vista window that looks something like this:



Fields are described from left to right below:

Connection Status	Made to look similar to a real 3270 terminal. The underscored **A** character appears when a connection is made. The leftmost character "M" changes to a "C" when screen is being captured to a file.
Terminal Status	Shows current terminal status (see table below)
Last Transaction Time	Time (in seconds and tenths) of last terminal transaction
Date	Current date MM/DD/YY
Julian	Julian day of the year
Time	Time in HH:MMxM - 12 hour time
Host Name	Host name of current connection
Macro Status	R recording, P playing, X paused
Copy Buffer	Number of the copy buffer currently being pasted
Shifter Key	Shows S, C, or A, for Shift, Ctrl, or Alt key being held down
Numeric Lock	N indicates cursor is within a numeric lock field
Caps Lock	Shows A or a indicating caps lock on or off
Insert Mode	Show ^ character if insert mode is active
Row, Col	Shows current cursor position on the screen. If row,col is separated by a slash, col shows current real column in ISPF editor.

Terminal Status Descriptions:

Disconnect	Terminal is not currently connected
Start	Connection is ready to start
Connecting	Terminal is attempting connection. When the attempt is near the end of the specified timeout, this status field counts down the number of remaining seconds.
Connected	Terminal has just been connected
X	Terminal is waiting for a host response
X SYSTEM	Terminal is waiting for a host response
	Input is inhibited at the current cursor location
	Input is inhibited because insert mode is active, and inserting more characters would result in a loss of data at the right of field

These show status during a data transfer, mostly these are only useful if the transfer fails and you need to know where it was at.

Tailoring the Status Bar:

The center text of the status bar, from Last Transaction Time to Host Name in the example above, can be changed by the user. Constant text and Variables can be defined in the string, which is automatically centered in the available status bar area. If the string is too large to display, the left and right edges are clipped as needed. Select the Options/Options menu item, and look for the Status Line field in the General options panel.

The default string supplied is:

```
&trantime %m/%d/%y.%j %I:%M%p &hostname
```

where:

&trantime	variable showing last transaction time
%m/%d/%y.&j	date in the format MM/DD/YY.DDD
%I:%M%p	time in the format HH:MMxM
&hostname	variable showing current host name

See System Variables for a list of available system variables, such as &trantime and &hostname.

See Date/Time Variables for a list of variables available for data/time substitution, such as %m and %d above.

Functions

Vista functions are divided up into the 5 groups shown below:

Action Functions

Action Functions cause instructions and/or data to be sent to the host. These functions include, Enter, PA1, PF1, Clear, Attention, and others.

Edit Functions

Edit Functions modify the screen locally, without sending any data or notification to the host. Examples of these include Home, NewLine, EraseEndOfField, etc.

Character Functions

These just type a particular character on the screen. Any character can be assigned to any key.

Operator Functions

Operator Functions include items such as SendFile, ExitEmulator, Printscreen, etc, that do not fall into any of the categories above.

Macros

Macros are *.MAC files in the Vista directory, and can of course issue keys and function calls as needed to do the work required - such as logon automatically to a host at startup. Macros can also be setup to do functions not available in any of the other types, such as calling WinExec to open a notepad window or view the clipboard.

The Vista functions can be executed by various methods, including:

 Keyboard or Mouse assignment using the keyboard editor

 Toolbar or Keypad button assignment

 From within a macro, using the Keys("....") function

...and in some cases, from the Menu Bar

See Keyboard Defaults and Toolbar Defaults for a description of how these user-modifiable items are supplied at installation time.

Action Functions

Action Functions cause instructions and data to be sent to the host. By default, most action keys are setup to be saved in the type-ahead buffer if issued when the screen is locked. The action keys Attention, and PA1-PA3 however, default to be issued immediately since we assume the user needs to interrupt the current process.

Attention	Sends an Attention Interrupt to the host
Clear	Clears the screen, and tells the host about it
Enter	Sends modified data to the host
PA1-PA3	Sends appropriate code and data to the host
PF1-PF24	Sends appropriate code and data to the host
SysRequest	Sends appropriate interrupt code to the host

Attention - Can be used to interrupt a currently running TSO command, for example, or signal the host for various other items. Attention sends a x'FFF3' (IAC BREAK) to the TN3270 server.

Default key - Escape

Enter - Sends an AID (Attention identifier) of x'7D' to the host, along with any fields on the screen the user may have modified.

Default keys - Right Ctrl, Ctrl-Enter, NumPad-Enter

Clear - Clears the local screen and also sends an AID (attention id) of x'6D' to the host, telling it that the screen has been cleared. Typically the host sends some data back or takes some other action.

Default key - Pause

PA1 through PA3 - typically used to interrupt the host in various ways. They send an AID of x'6C', x'6E', and x'6B', respectively to the host.

Default keys - Ctrl-Insert, Ctrl-Home, Ctrl-PageUp

PF1 through PF24 - typically used to control various activities on the host, and can change their function dynamically depending on the host application. They send an AID (attention id) to the host to indicate which key was pressed.

Default keys - F1 through F12, and Ctrl-F1 through Ctrl-F12 for PF13 through PF24
PageUp and PageDown keys are also set to F7 and F8 respectively, for ISPF scrolling.

SysReq - Sends x'F0' to the host, which is usually interpreted by the tn3270 server as a SysReq interrupt.

Default key - Shift-Esc

Edit Functions

Edit Functions modify the screen without sending any data or notification to the host. By default, almost all of these are set to be saved in the type-ahead buffer if typed on a locked screen.

BackNewLine	Move cursor to the leftmost input field on the previous line
Backspace	Move the cursor back 1 position
BackTab	Move the cursor back to the previous field on the screen
BottomHome	Move cursor to the lower left input field on the screen
CapsLock	This function enables the Caps Lock key
ClearBuffers	Clear all copy/paste buffers, including the Clipboard
Copy	Copy selected text to Clipboard
CopyAppend	Add selected text to the end of the copy buffer.
CopyFields	Copy with tabs between fields
Cut	Copy selected text to Clipboard and Delete
Delete	Delete single character or selected text
DeleteWord	Delete word at cursor position
Down	Move cursor down one line
Down2	Move cursor down two lines
Dup	3270 Dup key
End	Move cursor to end of text in input field
EraseEndofField	Erase text from cursor position to end of field
EraseInput	Erases all fields on the screen and marks them as not modified
FieldMark	Type a FieldMark
Home	Move cursor to the upper left input field on screen
Insert	Set or toggle insert mode, depending on Display option
Left	Move cursor to the left one character
Left2	Move cursor to the left two characters
LeftField	Move cursor to the left most input field on the current line
NewLine	Move cursor to the leftmost input field on the next line
NullKey	Set key to do nothing
NumLock	This function enables the Numeric Lock key
Paste	Paste Overlay or Paste Fields depending on Option settings
PasteContinue	Continue pasting from where we left off
PasteField	Paste data and repeat on any lines below cursor
PasteInsert	Paste data to the screen, moving existing data to the right
PasteJCL	Replace text based on screen content and mouse position
PasteOverlay	Paste data and overlay any input fields
PasteRepeat	Paste data and repeat on any lines below cursor
PasteWindow	Paste a long text string into a formatted rectangle
PopupMenu	Show a popup menu for Undo/Cut/Copy/Paste/Select All
Reset	Reset the emulator
Right	Move cursor to the right one character
Right2	Move cursor to the right two characters
SelectAll	Select all text on screen (including blank areas)
SelectField	Select current field
SelectJCL	Select text based on screen content and mouse position
SelectLine	Select current line

SelectLocation1	Start a Select Box at the current cursor location
SelectLocation2	End a Select Box at the current cursor location
SelectText	Select all text in the field pointed to by the mouse
SelectWinDown	Show a select window and move corner down 1 char
SelectWinLeft	Show a select window and move corner left 1 char
SelectWinRight	Show a select window and move corner right 1 char
SelectWinUp	Show a select window and move corner up 1 char
SelectWord	Select current word
Tab	Tab to the beginning of the next input field
Undo	Revert back to the last screen sent by the host
Up	Move cursor up one line
Up2	Move cursor up two lines
WordLeft	Move cursor left one word
WordRight	Move cursor right one word

BackNewLine - Moves the cursor to the leftmost field *above* the current cursor location. This is the opposite action of the Newline key.

Default key - Shift-Enter

Backspace - Moves the cursor one position to the left. If the Display option Destructive Backspace is set, the previous character is also removed from the screen.

Default key - Backspace; Default option - Destructive Backspace is Off

BackTab - Moves the cursor back to the previous field. Note, this field may be up and to the right of the current cursor location.

Default key -Shift-Tab

BottomHome - Moves the cursor to the lowest left field on the screen. This is useful for applications that insist on putting the command line at the bottom (SAA standard, need I say more?)

Default key - Alt-Home

CapsLock - Makes the capslock key function normally. However, you can edit the capslock key to something else (or disable it with the NullKey function) if you want.

Default key - Caps Lock (this function cannot be used with any other key)

ClearBuffers - Clears all the copy buffers, and also clears the clipboard

Default key - Alt-F12

Copy - Copies currently selected text to the clipboard and to copy buffer 1. All other active copybuffers (if any) are shifted right and the oldest buffer is lost. Text is selected by holding down the left mouse button, or by any of the Select* functions. If no text is selected, this function is ignored.

Default key - Ctrl-C

Default toolbar button is 

CopyAppend - Appends the currently selected text to the clipboard and to copy buffer 1, similar to the Copy function. If no text is currently selected, CopyAppend will select text that is on the screen using the previously used selection area. This is so you can do the following series of commands to copy a large amount of data to the copy buffer (up to 64K):

Use ISPF option 2 to edit a dataset
Scroll up 1 line to get the 'Top of Data' line off the screen
Use the mouse to select the entire visible edit text area
Press the Copy key (default Ctrl-C)
Move the cursor to the command line and press F8 to scroll down
Press the CopyAppend key (default Ctrl-A)
Repeat F8/Ctrl-A until all the data is copied or you fill up the 64K buffer

Default key - Ctrl-A

CopyFields - Copies selected text to the clipboard just like the Copy function, but **Copyfields** places tab characters (hex 09) between horizontal fields. This is useful when copying fields to an Excel spreadsheet, for example.

Default key - None

Cut - Copies the currently selected text to the clipboard and then deletes it from the screen. Deleted text is replaced with spaces or nulls depending on the Cut/Paste option Replace with Nulls or Replace with Spaces. If Nulls are used, the data to the right of the deleted area typically 'collapses' left since nulls are not sent back to the host. If no text is selected, this function is ignored.

Default key - Ctrl-X; Default option - Replace with Nulls

Default toolbar button is 

Delete - Deletes the character at the current cursor position, and moves any characters to the right of that location left 1 position.

If a selection window was drawn on the screen, the delete function will remove *all* text within the selection window. Selected text that is deleted is replaced with spaces or nulls depending on the Cut/Paste option Replace with Nulls or Replace with Spaces. If Nulls are used, the data to the right of the deleted area typically 'collapses' left since nulls are not sent back to the host

Default key - Delete; Default option - Replace with Nulls

DeleteWord - If cursor is on a word, that word is deleted and words to the right of the deleted word are scooted left as needed to replace the deleted text. If this function is used without the cursor on a word, then a single character is deleted.

Note: A 'word' is considered to be a string of characters bounded by spaces.

Default key - Ctrl-D

Down - Moves the cursor down one character position. If the cursor was at the bottom of the screen, it wraps to the top of the screen.

Default key - Down-Arrow

Down2 - Moves the cursor down two character positions. If the cursor was at the bottom of the screen, it wraps

to the top of the screen.

Default key - Alt-Down-Arrow

Dup - Types an overscore asterisk on the screen indicating the Dup character (x'1C'), and also tabs the cursor to the next field.

Default key - None

End - Moves the cursor past the last text in the current unprotected field. If this new position results in the cursor being on a protected field or field character, the cursor is tabbed to the next field on the screen. If this command is used while the cursor is in an unprotected field, the cursor is first tabbed to the next unprotected field, and then the End is executed.

Default keys - Ctrl-E, Shift-End

EraseEndofField - Replaces the character at the cursor position with nulls, and all characters to the right of that cursor position. The function only works in unprotected fields, of course.

Default key - End

EraseInput - Erases all input fields (to nulls) and marks them as not modified so data will not be sent to the host even if data was previously typed in those fields. In some applications, such as ISPF that "remembers" what was previously on the screen, this function can cause very unusual results.

Default key - none

FieldMark - Types an overscored semi-colon (x'1E) on the screen at the cursor position. This character is typically used to stack TSO commands together, among other things.

Default key - none

Home - Moves the cursor to the uppermost and leftmost field on the screen. Usually this is at the command line unless you've installed some of those *fine* SAA compliant applications.

Default key - Home

Insert - Sets the input mode so that any characters typed on the screen will scoot any existing characters over to the right instead of overlaying them. Note that once a field is full you cannot insert any more characters - Character keys will be inhibited.

The Display option Toggle Insert Key causes insert mode to be turned on and off with each press of the Insert key, like the typical Windows program usually does. But real 3270's don't reset insert mode until you press the Reset key (default left Ctrl) or press an Action key such as Enter.

Default key - Insert; Default mode: Toggle Insert Key is Off (acts like a real 3270)

Left - Moves the cursor left one position. If the cursor was at the left side of the screen, the cursor wraps to the right side of the screen.

Default key - Left-Arrow

Left2 - Moves the cursor left two positions. If the cursor was at the left side of the screen, the cursor wraps to the

right side of the screen

Default key - Alt-Left-Arrow

LeftField - Moves the cursor to the beginning of the leftmost field on the current line.

Default key - none

NewLine - Moves the cursor to the leftmost field on the next line with a field on it.

Default key - Enter

NullKey - Sets a key to do nothing

Default key - n/a

NumLock - Sets the Num Lock key active. This allows the Num Lock function, while still allowing you to edit the Num Lock key to something else (or disable it with NullKey)

Default key - Num Lock (this function cannot be used with any other key)

Paste - Pastes text from the current copy buffer to the screen at the cursor location. This function actually will do a PasteOverlay or PasteField function, depending on the Paste settings in the Cut/Paste option panel. See notes on those two functions for actual pasting logic.

Vista supports up to 9 (64K) copy buffers (default is set to 3). Doing a Paste the first time will paste buffer 1 (the contents of the clipboard) to the screen. Pressing Paste again with no other intervening activity will then paste buffer 2 on the screen, and so on, until the end of the copy buffers is reached or an empty buffer is seen. Paste then circles back to buffer 1 and continues until you find that data you thought you had lost. Pressing any key other than Paste ends the cycle and leaves the last copy buffer on the screen.

If you run into trouble, press the Undo (default Ctrl-Z) function and the screen should revert back to the last transmission from the host, assuming that is what you want.

Default key - Ctrl-V; Default options: Overlay Input Fields, Number of copy buffers = 3

Default toolbar button is 

PasteContinue - If the total contents of the buffer you previously pasted with PasteOverlay, PasteField, or PasteWindow did not completely fit in the designated area, PasteContinue will continue pasting data from where the previous command left off. Also, the previous command paste *style* (overlay, field, or window) will be used for any continued pastes.

Default key - Ctrl-B

PasteField - Pastes each line of input to each field on the screen. Each line will be pasted to the beginning of each field, regardless of the location of the cursor within the first field. Author's note: I'm still trying to find a use for this function.

See the Paste function for a description of multiple copy buffers, which also work with this function.
See PasteContinue for a description of how to continue pasting a buffer larger than the screen area.

Default key - None

PasteInsert - Pastes each line of input as if the clipboard buffer was a block of text at the cursor position. Existing data is moved to the right as needed. If this process would result in data loss at the right of the screen, the user is notified. This notification can be disabled if desired, on the Cut/Paste option panel.

See the Paste function for a description of multiple copy buffers, which also work with this function.
See PasteContinue for a description of how to continue pasting a buffer larger than the screen area.

Default key - Ctrl-Q (so you can easily hit this key with your left hand)

PasteOverlay - Pastes each line of input as if the clipboard buffer was a block of text at the cursor position which overlays a block of screen data the same size. Screen fields that are protected are not modified, of course, even if there happens to be data in the block at that location.

See the Paste function for a description of multiple copy buffers, which also work with this function.
See PasteContinue for a description of how to continue pasting a buffer larger than the screen area.

Default key - Ctrl-V; Default options - Number of copy buffers = 3

PasteRepeat - Pastes the contents of the current clipboard to the current cursor location, and then repeats the paste over and over again (moving down the screen) until the bottom of the screen is hit.
Author's note: This is more useful than it may appear at first.

Default key - Ctrl-R

PasteWindow - This function is designed to take an unformatted string (a string with no CRLF's in it, such as text from an e-mail window), and paste the text to a selected rectangular area on the screen. Text will be broken at spaces and comma's (like a word processor) to fit as well as possible within the selected area.

PasteWindow requires that you select the area to paste with the mouse (or macro) before pasting. Once this selection is done though, you do not have to respecify the area for any subsequent PasteContinue functions. You can, however, select a new area for PasteContinue if you want to, and, for example, make columnar data out of a long unformatted text string.

Default key - Ctrl-W

PopupMenu - If you want Vista to look more like a typical Windows application, you might want to edit your right mouse key to PopupMenu (although you really should examine the default SelectJCL function before doing this). PopupMenu pops up the typical menu in the middle of the screen that allows you to Undo, Cut, Copy, Paste, Delete, and Select All

Default key - Shift-Right-Mouse-Single-Click

Reset - Resets the keyboard. If you are running with the default Display option Auto Key Unlock on, then you won't be using Reset nearly as often as you did on a real 3270 terminal. However, with Auto Key Unlock off, you'll be needing it a lot.

Reset also sets the Insert key off, regardless of the Display option setting Toggle Insert Key

Default key - Left Ctrl

Right - Moves the cursor right one position. If the cursor was at the right side of the screen, the cursor wraps to the left side of the screen.

Default key - Right-Arrow

Right2 - Moves the cursor right two positions. If the cursor was at the right side of the screen, the cursor wraps to the left side of the screen.

Default key - Alt-Right-Arrow

SelectAll - Draws a select window around the entire text area, and waits for a Cut, Copy, or Delete function.

Default key - Ctrl-S

SelectField - Draws a select window around the 3270 field indicated by the mouse or macro row/col position. The field may be either protected or unprotected, and all field positions (including spaces or nulls) are included.

Default key - Ctrl-F

SelectLine - Draws a select window around the line indicated by the mouse or macro row/col position. The line may consist of either protected or unprotected fields, and all characters (including spaces or nulls) are included. SelectLine always selects a rectangle as wide as the screen.

Default key - Ctrl-L

SelectLocation1 - Marks the first corner of the select rectangle at the current cursor position, or the row/col indicated by a macro. Designed mainly for macro processing

Default key - None

SelectLocation2 - Marks the second corner of the select rectangle at the current cursor position, or the row/col indicated by a macro. Designed mainly for macro processing

Default key - None

SelectText - Selects all the text in the field pointed to by the cursor or mouse, but trims leading and trailing blanks.

Default key - None

SelectWinDown - Moves the lower right-hand corner of the select box down one position. Draws the selectbox on the screen if not already active.

Default key - Shift-Down-Arrow

SelectWinLeft - Moves the lower right corner of the select box left one position. Draws the selectbox on the screen if not already active.

Default key - Shift-LeftArrow

SelectWinRight - Moves the lower right corner of the select box right one position. Draws the selectbox on the screen if not already active.

Default key - Shift-Right-Arrow

SelectWinUp - Moves the lower right corner of the select box up one position. Draws the selectbox on the screen if not already active.

Default key - Shift-Up-Arrow

SelectWord - Draws a select box around the word at the current cursor location. If cursor is pointing to a space, this function does nothing.

Note: A 'word' is considered to be a string of characters bounded by spaces.

Default key - Ctrl-Right-Mouse-Single-Click

Tab - Moves the cursor to the beginning of the next input field on the screen. Note, on an unformatted screen (a screen with no fields), this command moves the cursor to the top-left position.

Default key - Tab

Undo - Resets the screen back to the way it was when the last transmission was received from the host, or if already pressed, does a 'Redo' to put the screen back the way it was at Undo time.

Default key - Ctrl-Z

Default toolbar button is 

Up - Moves the cursor up one position. If the cursor was at the top of the screen, the cursor wraps to the bottom of the screen.

Default key - Up-Arrow

Up2 - Moves the cursor up two positions. If the cursor was at the top of the screen, the cursor wraps to the bottom of the screen.

Default key - Alt-Up-Arrow

WordLeft - Moves the cursor left one word, regardless of if the field is protected or unprotected.

Note: A 'word' is considered to be a string of characters bounded by spaces.

Default key - Ctrl-Left-Arrow

WordRight - Moves the cursor right one word, regardless of if the field is protected or unprotected.

Note: A 'word' is considered to be a string of characters bounded by spaces.

Default key - Ctrl-Right-Arrow

The SelectJCL Function

SelectJCL is set as the default right-mouse-single-click function, because the author believes it should be helpful for selecting text items on the typical mainframe screen. Here is a quick description of what it can do for you:

SelectWord provides a pretty good way to select data for copy/cut/delete as long as the string is bounded by spaces. But to pick up fields smaller than a word you normally have to resort to the tedious and error-prone dragging of the mouse to create a select window.

SelectJCL may help select small items such as JCL parms, dataset names and portions of dataset names, and other similarly formatted data, by using the mouse position at select time as a factor in deciding what to select. For example, if you click at various spots within this JCL line (as indicated by the numbers and letters below), you'll get results as indicated in the table below:

```
//INPUT  DD  DSN=SYS1.TCPIP.LOAD(MEMBER1),DISP=SHR
             1 23 45 6 7 8  9  ab   c     d e  f
```

Click	Data Selected	Comments
1	DSN=SYS1.TCPIP.LOAD(MEMBER1)	returns the whole parm
2	DSN	right-half returns word
3	DSN=SYS1	mouse on special char
4	SYS1.TCPIP.LOAD(MEMBER1)	dsname only returned
5	SYS1	right-half return HLI
6	SYS1.TCPIP.LOAD	repeated special char
7	TCPIP.LOAD(MEMBER1)	convert to TSO dsname
8	TCPIP	word only
9	LOAD(MEMBER)	not much use, but...
a	LOAD	word only
b	(MEMBER1)	member with parenthesis
c	MEMBER1	member without parens
d	DISP=SHR	whole parm
e	DISP	word only
f	SHR	word only since at end

Some additional logic included in SelectJCL is:

Click a special character (non alphanumeric) that is surrounded by 2 non-special characters will return the words surrounding that character, plus any other adjacent words connected by that character. This allows us to pick out just the dataset name (without the member) in example '6' above.

Click on a left paren, and everything to the closing paren will be included, plus the closing paren. Same logic is used for ticked or quoted strings.

Click the left half of the first word within a ticked, quoted, or paren'd string and you should receive the data within the ticks/quotes/parens, but not the delimiters themselves.

Click the a field character (the blank right before a field) or any blank in the field, and the entire field will be selected.

Click on a special character preceded by a space, and the selected word (bounded by spaces) will be selected.

If the first character of a line is a forward slash (JCL indicator) and you've clicked on either the first or second character in the line, all the text in the line will be selected.

Click on a special character that has at least one special character next to it on either side, and the logic assumes you want a 'word' of special characters bounded by alphanumeric delimiters.

Selecting any space or null character will select the entire field.

Note:

Selection will not wrap beyond one line. For example, if you select the first tick of a quoted string, but the closing tick has wrapped to the next line (typical in syslog and JCL), the ‘smart’ selection fails and only the first tick character will be selected.

The PasteJCL Function

PasteJCL complements the SelectJCL function. Once SelectJCL has been used to select a screen item such as a JCL parm, Datasetname, Member, etc., you can use **PasteJCL** to replace similar items with the clipboard string. Please read the help information on SelectJCL if you haven't already done so.

An example of PasteJCL:

Suppose you have the JCL below as part of a sort job:

```
//SORTIN   DD  DSN=TEST.INPUT.DATA(MEMBER),DISP=SHR
//SORTOUT  DD  DSN=USER2.OUT.FILE,DISP=SHR
```

But today you want to sort the input dataset in place, so you'll need to copy the first dataset name down to the second DD card. Try this:

Move the mouse to the first part of the word TEST in the input dataset name, and right click to do a SelectJCL (or ctrl-J on default keyboard). This should select the entire dataset name, including the member name.

Now move the mouse to the first part of the word USER2 in the output dataset name. Hold a control key down and right click to do a PasteJCL function (ctrl K on default keyboard). The output datasetname should be replaced with the contents of the clipboard, scooting the DISP=SHR out to the right as needed.

The way PasteJCL decides what to replace is based on the SelectJCL logic previously described. As with SelectJCL, the mouse position and screen content determines what text gets selected for replace. Here's another example:

Original text:

```
//SORTIN   DD  DSN=TESTID1.INPUT.DATA(MEMBER),DISP=SHR
//SORTOUT  DD  DSN=USER2.OUT.FILE,DISP=SHR
```

This time, right-click the first part of the word INPUT. SelectJCL should select the dataset name, but without the high-level qualifier.

Now hold ctrl and then right-click the first part of the word OUT. PasteJCL will replace OUT.FILE with the clipboard data, leaving the USER2 high-level qualifier intact.

Some Restrictions:

PasteJCL can only paste a single line of text.

When replacing data with longer strings, data to the right of the string is moved to the right as necessary - sometimes even off the edge of the screen (where it is lost).

Characters

Character Functions simply type the specified character on the screen at the current cursor location. The keyboard, mouse, toolbar buttons, and macros, can all issue characters.

Any one of 224 printable ASCII characters can be specified. They are translated to their appropriate EBCDIC equivalent when sent.

Operations

Operations allow the user to assign functions like SendFile, PrintScreen, etc. to a key, toolbar, or call them from a macro. Operations include:

Capture	Toggle screen automatic capture to file
Cascade	Cascade all open Session Windows
CascadeFromHere	Cascade all open Session Windows from current position
DebugFile	Call up a previously saved debug file
Disconnect	Disconnect from the currently active host
EditKeyboard	Edit the keyboard layout and functions
EditKeypad	Edit the keypad layout and functions
EditProfile	Edit Session Profile
EditToolbar	Edit the toolbar layout and functions
ExitAll	Exit all open Sessions
ExitEmulator	Interrupt the session and exit the emulator program
FontLarger	Change to next Larger font
FontNarrower	Change to next Narrower font
FontSelect	Display Font Selection Dialog
FontShorter	Change to next Shorter font
FontSmaller	Change to next Smaller font
FontTaller	Change to next Taller font
FontThicker	Change to Vista Thick Fonts
FontThinner	Change to Vista Thin Fonts
FontWider	Change to next Wider font
HelpContents	Show the Help Contents page
HelpIndex	Show the Help Index
LocateCursor	Flash the cursor so you can find it on a busy screen
MaximizeAll	Maximize all open Sessions
MinimizeAll	Minimize all open Sessions Windows
MoveCursor	Move cursor to mouse position
MoveCursorEnter	Move cursor to mouse position, then press Enter
NewSession	Start a new terminal session, the same as the current one
NewSessionAsk	Start a new terminal session
NextSession	Select Next Session
Pause	Pause Macro recording (press again to continue)
Play	Playback a previously recorded Macro
PlayStartUp	Play the default StartUp Macro
PrevSession	Select Previous Session
PrintFile	Reprints pc$print.txt (downloaded from PCPRINT cmd)
PrintScreen	Print the screen
PrintScreenAsk	Print the screen, but show standard print dialog first
ReceiveFile	Transfer a file from Host to PC
Reconnect	End current session (if any) and attempt to reconnect
ReconnectAsk	End current session (if any) and ask for reconnect
Record	Begin Recording a new Macro
ResetKeyBuffer	Clear the keyboard type-ahead buffer
SaveScreen	Save the current screen to a file

SaveScreenAppend	Append current screen to a file
SaveSession	Save the current session profile
SendFile	Transfer a file from PC to Host
ShowAttributes	Display attribute characters as dots
ShowKeypad	Shows the Keypad at the current mouse position
ShowMenuBar	Show or Hide the main Menu Bar
ShowMaximize	Maximize the main window
ShowMinimize	Minimize the main window
ShowNormal	Show the main window as a normal (sizeable) window
ShowRestore	Restore the main window to it's previous setting
ShowToolBar	Show or Hide the Tool Bar
Stop	Stop Recording or Playing a Macro
TransferLog	Display the File Transfer Log window

Capture - After each screen is received from the host and the screen is unlocked, send the entire screen to the current file specified in the Save Screen to Disk option. This function toggles screen capture on and off. Capture is always turned off at Vista startup.

When capture is on, the leftmost character on the status bar changes from "M" to "C".

Default key - none

Cascade - Cascade all open sessions at the top left of the screen. Each session remains it's normal window size.

Default key - none

CascadeFromHere - Cascade all open sessions starting at the position of the current window. Each session remains it's normal window size.

Default key - none

DebugFile - Call up and display a previously saved *.dbg file. Useful for debugging screen image problems

Default key - none

Disconnect - Disconnect the current session from the host, and do not try to reconnect. Normally the Reconnect function is probably what you really need.

Default key - none

EditKeyboard - Call up the key edit dialog, allowing you to specify functions for any key on a standard 101 key keyboard, and also mouse key operations.

Default key - none

Default toolbar button is 

EditKeypad - Call up the keypad edit dialog, allowing you to add, remove, and re-arrange keys on the popup keypad.

Default key - none

EditProfile - Call up the profile edit window, where you can set various Vista emulator options.

Default key - Alt-Z

Default toolbar button is 

EditToolbar - Call up the toolbar editor, where you can drag and drop toolbar buttons to create various toolbars that meet your needs. Buttons are assigned functions and macros, as with the keyboard editor.

Default key - none

Default toolbar button is 

ExitAll - Disconnect and close any currently open Vista Session windows. If any of these have Ask Before Closing option set, you will need to reply first.

Default key - none

ExitEmulator - Disconnect and close the current Vista session. If the Ask Before Closing option is set, you will need to reply first.

Default key - Alt-F4

FontLarger - Find the next larger font, if any where both the height and width of the font are larger than the current font. If this function seems to have no effect, you are already at the largest font that will fit on the screen as a window. Try maximizing the window to get a bit larger font, if possible.

Default key - none

FontNarrower - If the current fontname has a font size with the current height but a little bit thinner, then change the font. If there is no thinner font at the current height, this function does nothing and gives no error message.

Default key - none

FontSelect - Open the font selection profile window where you can specify font attributes, and related screen windowing options.

Default key - none

FontShorter - Attempt to find a font that is at the current width, but a bit shorter. If not found, this function does nothing.

Default key - none

FontSmaller - Change font to the next smaller font, where both the height and width are smaller than the current font. If we are already at the smallest font, this function does nothing.

Default key - none

FontTaller - Change font to the next taller font, where the width is the same as the current font. If no taller font is

found, this function does nothing.

Default key - none

FontThicker - If Vista Thin Fonts are currently selected, change to Vista Thick Fonts.

Default key - none

FontThinner - If Vista Thick Fonts are currently selected, change to Vista Thin Fonts.

Default key - none

FontWider - Find the next font wider than the current font, while keeping the same height as the current. If a wider font is not found, this function does nothing.

Default key - none

HelpContents - Display the Help Contents page.

Default key - none

Default toolbar button is 

HelpIndex - Display the Help Index.

Default key - none

Default toolbar button is 

LocateCursor - Flash the cursor so you can find it on a busy screen. This is especially useful when using functions like the ISPF FIND command. The locator action can be set to blink or explode (see the cursor profile options).

Default key - none

MaximizeAll - Maximize all open Vista sessions. The "A,B,C" buttons on the default toolbar can be used to select one of the first 3 sessions, or you can add more buttons as needed. Also, the Window menu item contains a list of all active windows.

Default key - none

MinimizeAll - Minimize all Vista sessions. To call a session back up, use Window's functions such as the alt-tab key, the task bar, etc. Windows that are closed while minimized will come back up next time as a normal window (not minimized).

Default key - none

MoveCursor - Move the cursor to the current mouse position. Normally this function is set to the left mouse button, but you can also set this to a key (you will still need to move the mouse though).

Default key - left mouse single click

MoveCursorEnter - Moves the cursor to the mouse position and then simulates pressing the Enter key. This is very useful for ISPF screen menu items, and other screen 'hot spots'.

Default key - left mouse double-click

NewSession - Open a new Vista session window, using the attributes from the current session.

Default key - none

Default toolbar button is 

NewSessionAsk - Open a new Vista session window, allowing you to change sessions, host connection, and terminal model.

Default key - none

Default toolbar button is 

NextSession - Swap to the next open Vista session window. If no other windows are open, this function does nothing

Default key - Ctrl-F1, Ctrl-N

Pause - When playing a macro, this function allows you to temporarily pause processing. Press the play key or button to continue when ready. If you pause while in record mode, recording is temporarily suspended and you can type or execute functions as needed without recording them. Press the record button to continue recording. A blinking "P" on the status bar indicates the processing is paused.

Default key - none

Default toolbar button is 

Play - Play a macro. This function will display a list of *.mac files and allow you to choose which one you want to run. A blinking "P" on the startus bar indicates a macro is currently playing.

Default key - none

Default toolbar button is 

PlayStartup - Play the startup macro, which is specified on the general options window. If no startup macro is specified there, this function does nothing.

Default key - none

PrevSession - Swap to the previous open Vista session window. If no other windows are open, this function does nothing

Default key - Ctrl-F2

PrintScreen - Prints the current screen on the default window's printer. If you first draw a select box with the mouse, only the selected portion of the screen is printed. If the User Name field on the Print options panel is non-blank, that data will be printed as a header on each print page, to identify who the screen print belongs to.

Default key - Ctrl-P

Default toolbar button is 

PrintFile - Prints the contents of pc$print.txt, which is the file used for the PCPRINT host command. Also can be used to print other files if called from a macro - for example:
Key(“PrintFile”,”c:\autoexec.bat”)

Default key - none

PrintScreenAsk - Same as the PrintScreen function, but allows you to specify printer-related defaults such as printer name, portrait, landscape, etc. prior to printing.

Default key - none

Default toolbar button is  while holding the ctrl key down.

ReceiveFile - Display the IND$FILE data transfer screen, allowing you to receive a file from the host to your PC.

Default key - none

Default toolbar button is 

Reconnect - Disconnect and attempt to reconnect the current host session. This is useful for when there are line problems, etc., and your session becomes locked up for some reason.

Default key - none

Default toolbar button is 

ReconnectAsk - Allows you to change session files, host names, and terminal models for the current session. This function will of course disconnect and reconnect to the host when executed.

Default key - none

Default toolbar button is 

Record - Select or type in a new macro name and description, then begin recording keystrokes. A blinking "R" on the status bar indicates recording is in progress..

Default key - none

Default toolbar button is 

ResetKeyBuffer - When the terminal is locked and Type Ahead is set in the General Options window, keyboard and other operator input is stacked in the key buffer. When the host unlocks the terminal and waits for input, this key buffer is sent back as if you had just typed it.

ResetKeyBuffer clears the key buffer at any time, if you decide you really didn’t want to send that data to the host after all.

Default key - none

SaveScreen - Save the text from the current screen in the text file specified on the General Options windowe. By default, this is set to VISTA.TXT in vista directory.

Default key - none

SaveScreenAppend - Append the text from the current screen to the end of the text file specified on the General Options windowe. By default, this is set to VISTA.TXT in vista directory.

Default key - none

SaveSession - Allows you to save the current session as a new name, or as the current name, and also allows you to create a new icon for the session if desired. Note: Vista saves the current session attributes at various times during processing, and always when closing a session, so there is really no need to use this option unless you are going to create a new session file under a different name.

Default key - none

SendFile - Display the IND$FILE data transfer screen, allowing you to send a file from your PC to the host.

Default key - none

Default toolbar button is 

ShowAttributes - Display field control bytes on the screen as dots. This function might be useful when programming screen images. Issue the function again to turn the dots back off.

Default key – none

ShowMenuBar - Toggles the appearance of the Menu Bar on or off. Also, this function is available on the “System Menu” , i.e. click on the icon at the far upper left of the main Vista window.

Default key - none

ShowMaximize - Sets the current window to the maximum size allowed by the screen. Note: certain applications, such as the Microsoft Office toolbar and the Windows task bar will make the usable full screen area smaller than the actual screen size. This may prevent Vista from using a font that fills the screen as fully as possible.

Default key - none

ShowMinimize - Minimizes the current window as if the user had hit the Minimize button or system menu option. To restore the window, most likely you'll need to click the Vista button on the Windows task bar.

Default key - none

ShowNormal - Shows the current window as a normal, sizeable, window.

Default key - none

ShowRestore - Restores the current window to it's previous Normal (sizeable) or Maximized setting. Typically this function would be used from a macro to restore the main window after a ShowMinimize instruction was previously executed.

Default key - none

ShowKeypad - Displays the current Keypad on the screen at the mouse position. To modify the keypad, or modify options (such as to leave the keypad on the screen), right-click any keypad button, or select the **Options/Edit Keypad** menu item.

Default key - none

ShowToolBar - Toggles the appearance of the Tool Bar on or off. Also, this function is available on the "System Menu" , i.e. click on the icon at the far upper left of the main Vista window.

Default key - none

Stop - Stop any macro recording or playing. If you were recording, a message appears telling you how many lines were saved in the previously specified macro file.

Default key - none

Default toolbar button is 

TransferLog - Display the IND$FILE file transfer log.

Default key - none

Macros

Macros can be assigned to any key or toolbar button. Macros can be created by recording keystrokes or editing the macro text yourself. Also, some macros are supplied which do various simple functions:

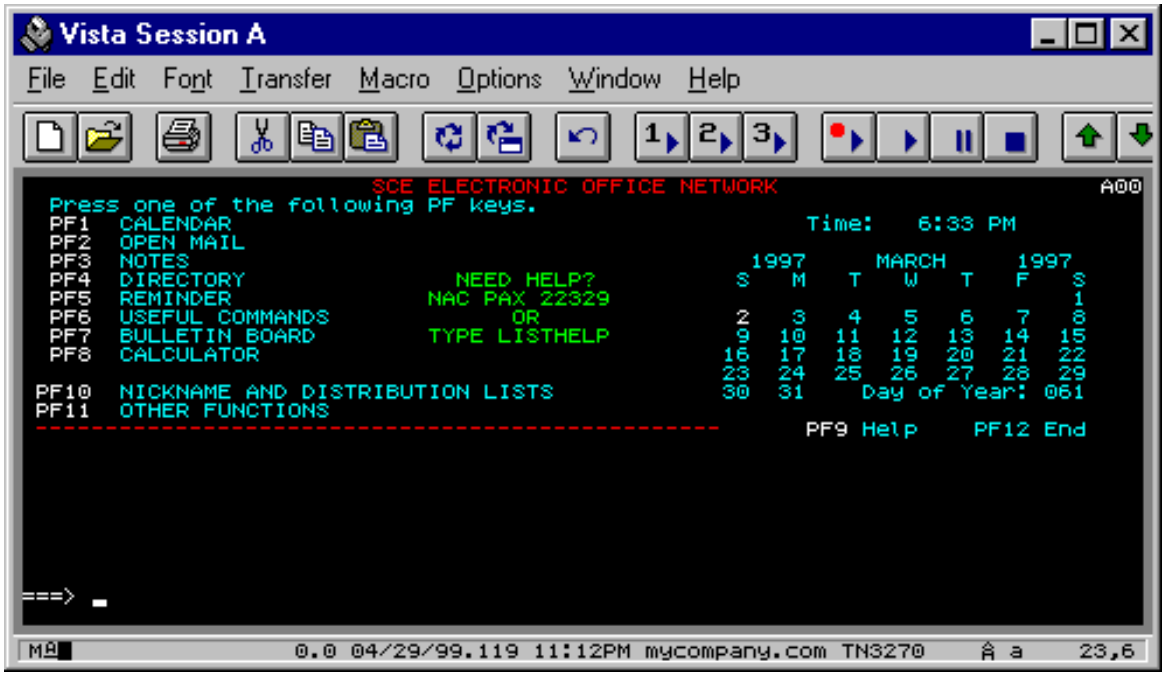
CLIPBRD.MAC	Calls up the windows clipboard
COPYFLD.MAC	Copies the field at the cursor position to the clipboard
COPYWORD.MAC	Copies the word at the cursor position to the clipboard
COPYLINE.MAC	Copies the line at the cursor position to the clipboard
COPYREP.MAC	Copies the current field to the clipboard, then pastes it and repeats it on any fields below the current cursor location.
HOTSPOT.MAC	Executes various functions based on screen contents at the mouse click location.
NOTEPAD.MAC	Calls up the windows text editor
SESSIONx.MAC	Swaps to another Vista session
USERx.MAC	Empty macros assigned to 3 buttons on the default toolbar

Once a macro is assigned to a toolbar button, you can hold the mouse over that button (with tooltips on) to see the macro description.

Also for toolbar button macros, you can right-click the toolbar button to bring up a popup menu with various macro related functions.

Hot Spots

Hot Spots in Vista are just text strings on the screen that cause certain action to occur. For example, you may see a screen like this in your daily work:



With hotspots enabled, you can click on the text to the left of each menu item (PF1, PF2, etc.) and Vista will issue the appropriate key function for you.

Hotspots are not enabled in Vista by default. I've found too many better uses for the mouse keys in my regular work, such as the SelectJCL and Copy functions set (by default) to the right mouse button. However, if you need hotspots, edit a mouse key function, such as double-right-click, to call the supplied macro "hotspot.mac". This will enable the default hotspot functions, making the PF strings active above.

You can also edit your own hotspot strings and functions into hotspot.mac as needed.

Options

Vista options are divided up into 8 groups:

General Options

General options, such as session file name, host name and port, keyboard and toolbar files, etc..

Display Options

Display related functions, such as terminal model, sound, blinking text options, and other key and display related parameters.

Cursor Options

Cursor and Ruler options.

Font Options

Font type and size, and window handling options.

Colors Options

Color definitions for all text types and styles, and other fields such as status bar and border colors.

Cut/Paste Options

Cut, Copy, and Paste options, such as the number of copy buffers, and copy box selection parameters

Print Options

Setup fonts, margins, and other items for screen prints, and the PCPRINT host command.

Miscellaneous Options

International code pages and keyboard layouts, and various file transfer options.

General Options

Session File - default standard.ses

Select an existing file, or type a new session file name. Session files contain a set of Vista options, such as screen colors, font name, and other parameters.

Session Name - default Standard Session

User assigned Session Name, which is assigned to the &SessName variable, and can be used in the Window Title, Status Line, or from within a macro.

Host Name - default none

Select or type a host name that will be used for this TN3270 session.

Window Title - default Session &w

User assigned text and variables that will appear in the top Title Bar for this session. Variables can also be included as needed. For example, the variable &Windowid is used on the default window title to show the session id letter (A through Z for up to 26 active sessions).

Status Line - default set to show date, time, and other information

User assigned text and variables that will appear in the bottom Status Bar for this session. Variables can also be included as needed. Click here for a description of the status line fields.

File for Save Screen to Disk - default vista.txt, overwrite

Select or specify a file name used to store captured screens. Also specify whether to **Overwrite** or **Append** to this file with each screen capture.

Edit Keyboard - default standard.key

Select or specify a keyboard file to be used for this session. Also, pushing the **Edit Keyboard** button calls up the keyboard edit dialog, where you can assign functions to keys as needed.

Edit ToolBar - default standard.bar

Select or specify a toolbar file to be used for this session. Also, pushing the **Edit ToolBar** button calls up the toolbar edit dialog, where you can assign buttons to the toolbar as needed.

Edit Keypad - default standard.pad

Select or specify a keypad file to be used for this session. Also, pushing the **Edit Keypad** button calls up the keypad edit dialog, where you can assign buttons to the keypad as needed.

Connect Timeout - default 10

Specify the number of seconds to wait before giving up when attempting to establish a connection.

Startup Macro - default none

Select or specify a macro to be used at the startup of this session (typically for logging on). If no startup macro is used, select **(none)** from the list

Show Window on Update - default off

When this option is selected, the Vista window will be activated and show itself whenever input from the host comes in, even if you are in some other Windows application at the time.

Ask Before Closing - default off

When selected, Vista will ask you if it's ok to close an active session

Show Tool Tips - default on

Shows a small yellow window with the tool button function or macro title, when the mouse is held over a button for a moment.

Change Icon

Press this button to change the icon for this particular session.

Auto Reconnect - default on

Attempt to reconnect when session is disconnected by a host logoff or another reason such as a phone line disconnection.

Use Alt for Menu - default off

If you are in the habit of using alt-x keypresses to activate Menu Bar functions, then try enabling this item. It will, however, prevent Vista from being able to process Alt key functions that might be edited in the keyboard map.

Display Options

Terminal Model - default Mod 2

Select the 3270 model screen size:

Mod 2	24 x 80	Standard
Mod3	32 x 80	Nice font on 1024 x 768 full screen
Mod4	43 x 80	Great for editing 80 column text
Mod5	27 x 132	Nice for sysout viewing
User	Up to 72 x 200	Even nicer for sysout viewing if you have a big screen TV

Sound - default all sound off

Alarm	Beep when an alarm is issued by the host program
KeyClick on Error	Click on input error, such as if you are attempting to type on a protected field.
Update Notification	Beep whenever the screen is updated by the host
After xx Seconds	Only sound the Update Notification if the transaction time has exceeded the specified time. This will interrupt you when a long transaction completes, without bothering you on the short ones.
Set Sounds	Press this button to assign WAV files to any of the sound events listed above. You can also add your own sounds by copying your *.WAV files to the Vista directory. For some additional sounds, check the web page at: http://www.tombrennansoftware.com/other.html

Blinking Text - default italic

Blink	blink the actual text on the screen. This is the default, but will drive you crazy soon.
Show as Italic	display blinking text as italics, and do not blink
Normal Text	ignore blinking text and just display it normally

Auto Keyboard Unlock - default on

A real 3270 keyboard locks if you attempt to type into a protected field, among other things, and requires the Reset key to be pressed before typing can continue. With this option set (the default) you don't need to press Reset, you just need to move the cursor to an unprotected field and continue typing.

Reset Insert on Action Key - default on

A real 3270 keyboard resets (disables) Insert Mode when any action key is pressed. This is

also the Vista default. However, if you want the Insert Mode to remain on between action keys, disable this option.

Toggle Insert Key - default off

A real 3270 keyboard resets Insert Mode only on action key or reset, not by pressing the insert key again. This option, however, allows you to toggle the insert key more like most PC programs do. The default (off) is the way the author likes it though.

Row/Col on Status Line - default on

Shows the current cursor position as row,col in the lower right corner of the window.

Modify Col in ISPF Edit - default on

When editing text in ISPF, this option will modify the column indicator to reflect the actual column in the dataset the cursor is at, rather than it's position on the screen. I believe this currently only works with the standard ISPF V4 edit, browse, and view screens.

Prefix Input with Spaces - default on

On a real 3270 (and most emulators), blanks to the right of text are actually hex zeros (nulls) although they appear as spaces. If you type within these null characters, any nulls to the left of your typing are not sent to the host, which results in your text 'collapsing' to the original text at the left. With this option set, Vista replaces any nulls to the left of your new text with spaces, avoiding this problem - even with NULLS ON in ISPF.

Convert Nulls to Blanks - default off

Rather than Prefix Input with Spaces, your application might call for all nulls (including those to the right or your typed text) to be replaced with spaces for some reason. This option supplies that action, although it has been known to cause problems when typing commands at TSO READY mode, since a large amount of trailing spaces may be sent to the program as parms.

Enable Auto Skip - default on

Enables the cursor to auto-skip to the next input field, if the program calls for it. This option defaults to enabled, but the author actually prefers it disabled in many cases.

Skip at End of Field - default on

When you are typing and you reach the end of a field, this option causes the cursor to jump to the beginning of the next field, as long as there are only field characters between the previous and next field. Useful for screens that have multiple fields on the same line with no intervening protected field headings.

Move Cursor on Activate - default on

When you have multiple Vista windows open, only one is active at a time. Clicking the mouse in the screen area of an inactive Vista window causes the mouse to be repositioned when this option is set on. If set off, the first mouse click in an inactive Vista screen area will **not** move the cursor - instead it remains where it was when the window became inactive.

All Upper Case - default off

This option must be a holdover from the dark ages of mainframe computing, although surprisingly the author still sees operators setting it on their real 3270 terminals.

Type Ahead - default on

Real 3270 terminals don't let you type anything while waiting for the host, and if you try, the status area shows that mean little man with his arms out telling you that not only did your typing get lost, but now you have to press the reset key. All this might be helpful for people who do data entry, but it's a bother for the typical mainframe user.

Enable Numeric Field Lock - default on

Programs requiring only numeric data may possibly set a field so that is impossible to type anything but numbers there. This option allows the user to override this and type whatever they want in the field.

Toggle Shift/Caps - default off

When the CapsLock key is active, the keys a-z will be shifted as usual to A-Z. In this state when the Shift key is used, Vista leaves the keys shifted to capitals so you don't accidentally type things like "hELLO, i AM tOM bRENNAN" . However, if you would like this capability for consistency with other Windows programs, set this option on.

Cursor Options

Normal Cursor - default horizonal, 3 lines, no blink

The normal (non-insert-mode) cursor can be set to be either a horizonal or vertical bar, and the size can be set to 1, 2, or 3 scan lines, or also set to a full or partial block cursor.

The Backspace function for non-insert-mode can be set to:

Move Left	Moves the cursor left just like the left arrow would do. Cursor can be backspaced across protected fields, and does not destroy or delete any characters.
Delete	Moves the cursor left, but only within unprotected (input) fields. Deletes the character that the cursor falls on. Characters to the right of the cursor are moved left.
Destruct	Moves the cursor left, but only within unprotected (input) fields. The character that the cursor falls on is "destroyed" (i.e. set to a space)

Insert Cursor - default vertical, 2 lines, no blink

The Insert Cursor is shown when you have pressed the Insert key. You can select this cursor to look just like the Normal Cursor, or set it to any of the styles described above.

The backspace function can also be set here as described above, and in this case will only take effect when we are in insert-mode.

Cursor Color - default - reverse video

This option gives you some control over the cursor color. Options are:

Reverse Video (default)

Cursor is draw with the opposite colors of those on the screen.

Character Color with Black Text

The Cursor changes to the color of the character it is sitting on. The text within the cursor is always black. This is useful if you need to know the color of the character cell while typing.

Field Color with Black Text

Similar to the option above, but instead of using the character color, the Field color is used, which in some cases might be different than character color (for example, in ISPF edit with HILITE ON).

White with Black Text

Cursor is always drawn white with black text, regardless of the character it is on, and regardless of other screen colors. If you are using the default black background color, this

option may make certain character colors (such as red), appear more readable when the cursor is on top of them.

Black with White Text

Similar to the option above, but for use when you have changed the default black background color to a lighter color or white.

Ruler - default none

The ruler can be set to a horizontal, vertical, or crosshairs (both) set of lines that follows the cursor, making it easier to line up text in some situations.

Locator Cursor - default explode

The LocateCursor function simply blinks or explodes (makes the cursor larger then smaller) so that you can find it on the screen if it becomes lost in that maze of sysout or text you are looking at. This is especially useful after issuing host Find commands which reposition the cursor to the found text.

Font Options

Font Name - default Vista Thick Fonts

Select a font for the emulator from the list, which should show any fixed pitch fonts on your system, and should include the 3 Vista fixed pitch fonts. GE fonts are actually used for screen graphic characters, so most likely you don't want to select that one.

Thick and Thin fonts are nearly identical except that the Thick ones are bolder in most cases, and show up better on monitors larger than about 15 inches. For smaller monitors (or for low resolution modes) you may find the Thin fonts easier to read.

True-type fonts are not supported, since they usually result in difficult-to-read generated graphics.

Size Window to Font - default on

When a window is resized either by you or by a host terminal model switch (known as an Alternate/Default switch), the emulator window size is adjusted accordingly to surround the resulting text size, although this may differ a bit from the actual window size requested.

Keep Font Constant on Mode Switch - default on

When an Alternate/Default switch occurs, for example, when the host switches the mode from Mod4 (43 lines) to Mod2 (24 lines), the screen or font will need to resize. With this option on, the font does not change at a mode switch - instead the window size adjusts to meet the new 3270 format. With this option off, the font changes in an attempt to keep the window size as close to the previous size as possible.

This option is not available when Size Window to Font is off.

Border Size - default 4

When a Vista session is "Windowed" (i.e. not full screen), a border of the specified size will appear around the screen text area. You can use this simply as an area to keep the text characters from butting right up against the edge of the window, or you can make the border a bit larger and change it's color to indicate which session you are in (great for that Y2K or Disaster Recovery test machine).

Font and Screen Example

Your selected font and estimated screen layout shows in these windows.

Font Nudge Buttons

If you need to make the font just a bit different from the current font, you can try the font nudge buttons, Smaller, Larger, Narrower, Wider, Shorter, and Taller. For example, if you are at an 8 x 10 font and want to go a little wider, you can push the Wider button to go to 9 x 10 font. But pressing the Wider button again will not result in any change, since there is no 10 x 10 font available. 9 x 10 is as Wide as the 10 pixel high font can get.

Colors Options

Group and Text Type

These 2 lists work together to allow you to select the different text and graphic entity types available on the Vista emulator.

Bright / Dark

This slider controls the brightness of all Vista colors at once, like the brightness control on a monitor.

Mid

Sets the Bright/Dark slider back to the middle

Colors

This is a list of 16 colors that can be active on the emulator at any one time.

Modify Colors

These scroll bars and numerical values can be set to choose any of 16M colors available on a standard Windows setup. Stick with the basic colors as much as possible though when using the emulator on 256 or less color installations.

How to Change Colors

To modify the color of a particular type of host text, first select the Group, and the Text Type within that group. The current assigned text Color will be shown by a black rectangle on the Color display. Choose the new color with the mouse, modify the color as required, and then press the Set Text Color button to actually set the color. You can also do the same for background colors by pressing the Set Background Color. An easier way to change the background for all colors at once though, is to simply modify the Black color itself.

Note: if you don't press either the Set Text Color or Set Background Color buttons, no changes will be made.

Cut, Copy, and Paste Options

Cut/Delete - default replace with spaces

Options here allow you to replace deleted text with nulls or spaces. Remember, nulls are not sent to the host - fields to the right of the deleted text will typically collapse to the left when updated by the host.

Copy

Vista supports multiple cut/paste buffers, and the number of buffers can be set here, from 1 to 9 (default is 3).

When any cut or copy operation is performed, the data is put into Copybuffer 1, and also into the clipboard. The previous contents of buffer 1 is copied to buffer 2, etc.

When a paste operation is done, Copybuffer 1 is displayed on the screen. If the same paste operation is done again with no intervening functions, Copybuffer 2 will be displayed, and so on.

Copy Box

The option here allows you to specify how many character cells the mouse must move when holding the left mouse button down, before starting to draw the selection box. Setting it to a number such as 2 (the default) will prevent the selection box from being drawn accidentally, such as when you are clicking to move the cursor.

Paste - default pasteoverlay, notify on

The Paste command actually does a PasteOverlay, a PasteField, or a PasteInsert,depending on the settings available here.

If Notify user on Insert Error is checked, a message box will appear if you do a PasteInsert which would cause a loss of text at the right portion of the screen.

The Move cursor to end on Paste option causes the cursor to move to the right of any pasted text. Note that the cursor will not be moved if the text buffer to be pasted contains more than one line. Also, the cursor will not be moved if the right portion of the pasted text is truncated by the right side of the screen.

Print Options

For each terminal model (2 through 5) and for the PCPRINT host command, you can specify various page formats, margins, fonts, and other items such as two-up printing capability.

Select the terminal model first, and then view or adjust other parameters which will be used for that terminal model (or PC Print) only.

Drag blue lines to adjust margins

Drag the lines to adjust how your final screen print will look. Rather than struggle with font size selection, these margins will adjust the font size for you *automatically*, to fit within the specified area on the paper.

Font - default Courier New

Select the font to be used. Be careful when specifying something other than Courier New, since other fonts may print some strange results. If all you want to do is change the font size, use the blue lines to adjust the margins instead.

User Name - default blank

Whatever you type here (usually a user name) will appear at the top of any screen print page. Useful for identifying the owner when sending data to a public printer, without wasting a separator page.

Note that variables can be used in this field, such as %c to show the date and time each page was printed.

Printer - normally set to "<default>"

Normally Vista sends screen prints (via the PrintScreen and PrintScreen Ask functions), or file prints (via the PrintFile function) to the Windows default printer. With this option though, you can force Vista to send print output to other than the default.

Mode - defaults to current terminal mode

Select the terminal model (or PC Print) to view or change options that will be used when printing in that mode.

Layout - all set to Window's default orientation, single page prints

Select Portrait or Landscape to force the page to that specific orientation. Or select Default to use the orientation currently selected in Windows for that printer.

Select Two Up to force 2 screens or PC Prints on a single page. **Note:** when this option is in effect and you want a single screen print, simply issue the Print function twice with no other intervening operations.

Highlighting - default set to "hilite"

Vista prints in regular and bold text, depending on this setting and the intensity of the text on the screen. The settings are:

hilite - prints intensified text as bold text, and the rest as regular text

regular - prints all text as regular (non-bold) text

bold - prints all text as bold text

Print

Press the Print button (over to the right) to print the current screen. Or if mode PC Print is selected, the last printed pc$print.txt file will be reprinted.

Miscellaneous Options

International - default United States (well, that's where I live!)

Screen - selects the code page used for screen I/O and file transfers.

Keyboard - changes the keyboard layout for a particular country, and also allows 'escape' code processing for special accented characters.

A note from the author about International options:

I believe the Screen code pages to be correct in most cases, but the Keyboard options may not always work correctly in your country. Unfortunately I don't currently have access to the various hardware and software versions used in various countries, so I'm just relying on user feedback at this point. Please contact me if you have trouble in this area. tom@tombrennansoftware.com

Also, if you are brave you might want to take a look at the country.lst file, which supplies the information for all code page, keyboard, and file transfer translations. I have no public documentation on this yet, so just contact me if you are interested.

Transfer - default ind$file, DOS code page, listcat, listds, list, 2500 bytes per block

These options allow you to supply a different command name when using Vista to transfer files, extra transfer parms if desired, a choice of DOS or Windows code page, commands for dataset listings, and WSF-mode blocksize.

Transfer Program

Indicate the transfer program you want to use for the file transfer. Most likely, this should be set to IND$FILE.

Send/Receive Options

IND$FILE transfer has various options, which are described below. But please remember that the following list was obtained by peeking at the IND$FILE load module, since I have found no documentation on it anywhere. The parameter descriptions below are mostly guesses based on the parameter name. Most should probably be issued as *name(parameter),* but who really knows?

Name	Parameter
APPEND	Append to the end of the host dataset
ASCII	Convert to/from ASCII on transfer
AVBLOCK	Unknown by me
BLKSIZE	Block Size
CRLF	Add/Remove CRLF characters on transfer
CYLINDERS	Indicate new allocation is specified in cylinders
FORMAT	Unknown by me
LRECL	Logical record length
NBR	Unknown by me
NEW	Create a new dataset
NOTRUNC	Do not truncate trailing spaces on transfer
PRIMARY	Primary space allocation

RECFM	Record format
SECONDARY	Secondary space allocation
SPACE	Indicate new allocation number of tracks or cylinders
TRACKS	Indicate new allocation is specified in tracks

PC Codepage

Set to either DOS or Windows, depending on the target PC environment. Note: most other emulators seem to use a DOS codepage when downloading, so you may want to stick with the default DOS translations for compatibility.

TSO Dataset List

Specify the command to be used when the **List** button is pressed on the transfer window, to retrieve dataset names from the TSO host. Note: Vista expects the response to be in the standard IDCAMS output format, so you obviously can't replace this with some other dataset list function. However, you can modify the command for users with their TSO profile prefix set, by specifying a command such as **LISTCAT LVL(MYPREF)**, for example.

TSO Member List

Specify the command to be used when the **Members** button is pressed on the transfer window, to retrieve a list of members within the selected PDS dataset. Note: as with the Listcat command, Vista expects a certain output format from the command.

CMS Dataset List

Specify the command to be used when the **List** button is pressed on the transfer window, to retrieve dataset names from the CMS host.

Transfer Blocksize

Specify the blocksize to be used for WSF (Write Structured Field) mode file transfer.

Changing the Vista Icon

Icons and Border Colors

Pressing the **Change Icon** button on the General options tab brings up a window where you can change the Vista window icon.

The first eight icons are simply color changes, which can help identify different windows. By default, the terminal window border is also changed to a color matching the colored icons. This border can also be made wider if desired (see the Border Size option on the Font option tab). On some full screen sizes though, the border area may disappear completely.

The other icons are “Vistas” that you might want to use if you get bored. There is no screen border color associated with these.

Restrictions

When you have multiple Vista windows open that use the same session file parameters, there may be some confusion when setting the icon (or for that matter, any other session file parameters). Basically under these condition, the following things happen:

Parameter changes will affect only the window where you made the changes. Other open windows will keep their old parameters.

Parameter changes will be saved in the session file, which is common to more than one open window. This means that any new windows opened with that session file will contain the newly updated parameters.

The last window closed will save the session file parameters, even though that last window might not have the parameters you really wanted saved. This may be confusing.

For these reasons, when using different icons or window border colors, it’s best to create multiple session files. Just use the **File/Save Session As...** function, which has an option to alter the icon.

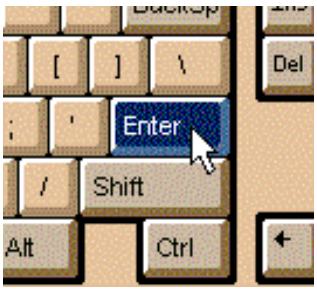
For example, at my desk I have standard.ses with the default gray icon and screen border, and I also have y2k.ses which points to our special Y2K test machine host name, with a purple icon and large purple screen border to let me know I’m on that machine.

Keyboard Editor

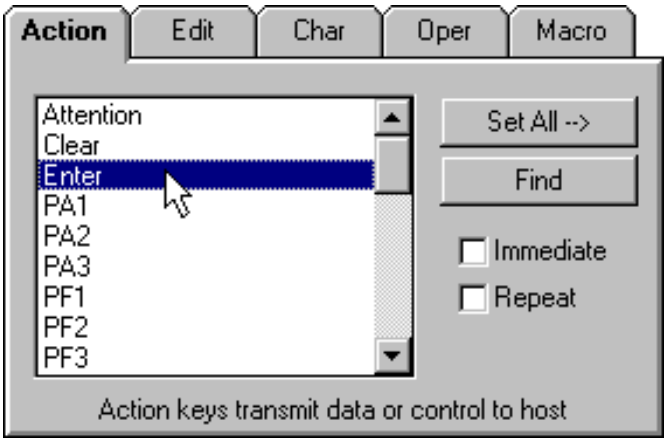
If you just want to find a key function, please see Keyboard Defaults for a list of Vista keyboard default key settings. Otherwise, follow the instructions below for editing your own keyboard layout.

How to Edit a Key

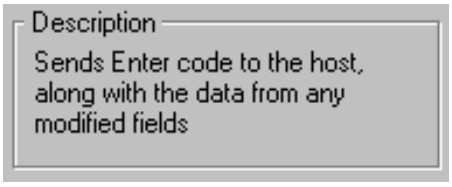
After starting the **Keyboard Edit** dialog (from the **Options** menu), select the key you want to modify such as this example where we want to change the Enter key from **Newline** to **Enter**:



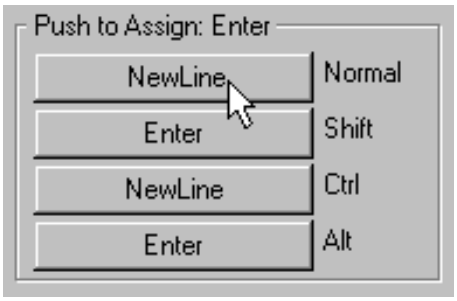
Then select the function you want the key to perform, using the tabbed Function Area list. In the example below, we've selected the **Action** tab, and then selected list item **Enter**.



Notice when selecting a list item, the **Set All** and **Find** buttons become enabled, and the Assignment Area title changes from **Push to Show** to **Push to Assign**, indicating we are ready to assign a key. Also, the Description Area changes and gives you a quick idea of what the selected function does.



To actually assign the selected function to the selected key, press one or more **assignment keys** in the Assignment Area. In the example below, we assign the **Normal** Enter key (i.e. no shift keys held), which was previously set to **NewLine**:



At this point you can continue editing other keys, and when finished, press the **Ok** button to exit the Keyboard Edit dialog. It will ask you if you want to save the key changes for future use, or use them for this session only.

Key Editing Details

Editor Modes

The Keyboard Edit dialog actually has **2 modes**, as indicated by the title in the Assignment Area.

When the title is **Push to Show**, and the **Set All** and **Find** keys are disabled, any key you select on the keyboard will cause the other screen areas to update and show you the current key assignment. You may also need to press the appropriate Assignment Area key (Normal, Shift, Ctrl, or Alt) to view details about that particular key, such as the Immediate or Repeat status.

When the title is **Push to Assign**, any button pressing in the Assignment Area will result in the selected function being assigned to the selected key.

To get into **Push to Show** mode, simple press any of the tabs in the Function Area. To get into **Push to Assign** mode, select any item in a Function Area list.

Immediate

With this option checked, a key will clear the type-ahead buffer, losing keys that were typed while the screen was locked, and then send this one key to the host immediately, regardless if the screen is locked or not. This is most useful for keys that interrupt processing, such as the Attention and PA keys.

With this option unchecked, if this key is typed when the screen is locked it will be sent to the type-ahead buffer (if enabled), and stored until the screen becomes ready for more input.

Repeat

With this option checked, a key repeat when held down. Unchecked, the key will not repeat if held, which is useful for Action keys such as Enter and the PF keys.

Note: When changing only the Immediate or Repeat flags of a particular key, you will first need to re-select the current key function so that the **Push to Assign** mode goes into effect. Then change Immediate or Repeat as required, and press the assignment buttons as usual.

Special Keys

Vista is unique among most emulators in the fact that you can edit ***any*** key on the standard 101 key keyboard, including such keys as Print Screen, Caps Lock, Num Lock, etc. Also, the Shift, Ctrl, and Alt keys (*Shifter* keys) can be edited as required, while still maintaining their original function. There are some restrictions with these special keys though:

Keys such as Escape and Tab have special Windows functions if certain shifter keys

are used with them. So when editing these keys you'll notice some buttons in the Assignment area are not available.

The shifter keys (Shift, Ctrl, and Alt) can be edited to one assigned function, which is triggered if the shifter key is pressed and released without pressing any other key. For these keys, the assigned action does not occur until the key is released. The Print key also falls into this category.

Also, because the shifter keys need to maintain their original shifter function, they cannot be set to repeat.

The Mouse

The mouse keys (double or single click, and in combination with shifter keys) can be edited to anything a normal key can be edited to. There is one thing to be aware of however: When double clicking the mouse, the assigned single click function also executes just prior to the double click function.

For example, in the default setup the left single click will move the cursor to the mouse location. The left double click will move the cursor and press Enter. With this combination, there is no trouble.

Other Restrictions

Vista of course supports editing a key so that it can be used in combination with any shifter key (shift, ctrl, or alt), but *multiple* combinations of shifter keys are not supported. For example, **alt-A** is supported, but **shift+alt+A** is not supported. You will instead get the action of the *last* shifter key you pressed.

Although keys such as PrintScreen, Alt, Capslock, and Numlock can be changed in Vista, you should be aware that using these may in some cases interfere with other Windows processing. For example, the PrintScreen key in windows copies a picture of the desktop to the clipboard. If you edit the Printscreen key to do something (like print a screen), the key still replaces the clipboard contents. You may just want to stick with the default ctrl-P for screen printing.

Keyboard Defaults

These are the default keyboard settings as provided by Vista at installation time. Of course you can change them to anything you want. See the Keyboard Editor for information on how to do this.

Obvious key functions, such as F1 or Home, are not listed below since they match the keyboard itself.

Default Keyboard Layout

3270 Function	Key
Attention	Escape
BackTab	shift-Tab
"Cent" Sign	shift-6
Clear	Pause
Enter	Right Ctrl key, and ctrl-Enter
Erase End of Field	End
F1-F12	F1-F12, also pageup and pagedown (F7,F8)
F13-F24	cntl-F1 to cntl-F12, or shift-F1 to shift-F12
NewLine	Enter key
"Not" key (for clists)	ctrl-[(left bracket)
PA1	cntl-Insert
PA2	cntl-Home
PA3	cntl-PageUp
Reset	Left Ctrl key
SysRequest	shift-Escape

Default Vista Function Keys

Vista Function	Key	Description
BackNewLine	shift-Enter	Moves cursor up one line and to the leftmost field
BottomHome	alt-Home	Move cursor to bottom command line
ClearBuffers	alt-F12	Clear all copy buffers
Copy	ctrl-C, Right Double Click	Copies selected text to clipboard
CopyAppend	ctrl-A	Append selected text to current clipboard
Cut	ctrl-X	Copies selected text to clipboard and then deletes it
DeleteWord	ctrl-D	Delete a word and scoot remainder of line to left
EditProfile	alt-Z	Calls up the profile edit dialog
End	ctrl-E or shift-End	Move cursor to end of a line of text
Exit Emulator	alt-F4	Standard windows exit key
MoveCursor	Left Single Click	Move cursor to mouse location
MoveCursorEnter	Left Double Click	Move cursor and press Enter
NextSession	ctrl-F1 or ctrl-N	Swap to next open Vista window
Paste	ctrl-V	Pastes selected text from clipboard to screen
PasteContinue	ctrl-B	Continues pasting text from the point we left off
PasteJCL	ctrl-K Right Single Click with Ctrl held	Paste clipboard to screen, replacing existing text depending on mouse position and screen content
PasteInsert	ctrl-Q	Paste to screen, inserting into existing content
PasteRepeat	ctrl-R	Paste to screen and repeat until bottom of screen
PasteWindow	ctrl-W	Paste clipboard into specified area on screen
PrevSession	ctrl-F2	Swap to previous open Vista window
PopupMenu	Right Single Click	Show Edit popup menu

	with Shift held	
PrintScreen	ctrl-P	Print current screen to default printer
SelectAll	ctrl-S	Draw a selection box around entire screen area
SelectField	ctrl-F	Draw selection box around entire 3270 field
SelectJCL	ctrl-J, Right Single Click	Select items depending on mouse position and screen content.
SelectLine	ctrl-L Right Single Click with Alt held	Select entire line of text
Undo	ctrl-Z	Puts screen back to last host transmission
WordLeft	ctrl-left-arrow	Move cursor one word to the left
WordRight	ctrl-right-arrow	Move cursor one word to the right

Toolbar Editor

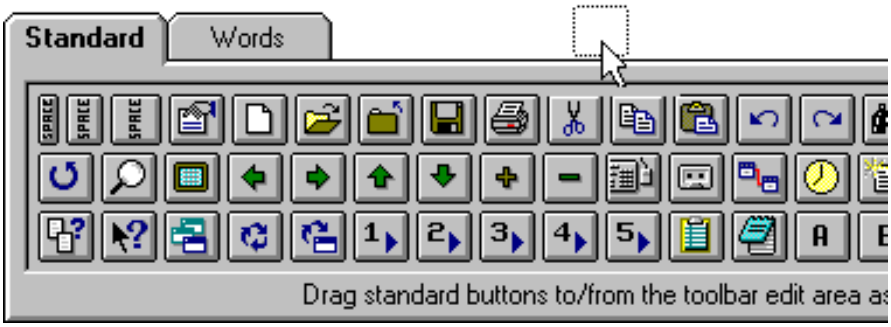
If you just want to find a toolbar function, please see Toolbar Defaults for a list of Vista default toolbar settings. Otherwise, follow the instructions below for editing your own toolbar layout.

Note: The Toolbar can be hidden if desired (usually to gain additional screen space).
Press the **Hide ToolBar** option on the "System Menu" (click the icon at the far upper left corner of the Vista main window).

How to Edit a ToolBar Button

After starting the **Toolbar Edit** dialog (from the **Options** menu), either select the button you want to modify in the ToolBar Edit Area, such as shown below:

... or drag a new button from the Standard button area to the ToolBar Edit Area before modification, as shown here:



Then edit the ToolBar function as you would any key on the keyboard. See Keyboard Edit if you don't already know how to do this. One interesting item to note here is that ToolBar buttons can change their function depending on if you are holding one of the shifter keys (shift, ctrl, or alt) or not.

Also, select the **Words** tab to select buttons with text descriptions for items such as PA1, Enter, and other functions.

ToolBar Edits

Adding Buttons

As shown above, simply grab a button from the Standard or Words area and drag it to the ToolBar Edit Area in the desired location.

Moving Buttons

Simply grab the button in the ToolBar Edit Area and move it to the position you want. When moving it a long distance, you may need to drop it, scroll the edit area, and then continue moving it. Buttons in the Standard or Words area cannot be moved.

Removing Buttons

To remove a button from the ToolBar Edit Area, grab the button and drag it back to the Standard or Words area. It will then be removed from the toolbar.

Adding Space Between Buttons

Drag one or more of the **Space** buttons in the Standard area and place them between buttons in the ToolBar Edit Area.

Restrictions

The Repeat option is not allowed on ToolBar buttons, so it is not shown.

Toolbar Defaults

These are the default toolbar settings as provided by Vista at installation time. Of course you can change them to anything you want. See the Toolbar Editor for information on how to do this.

Note: The last 5 toolbar buttons are not visible on a 640 x 480 screen.

Icon	Function	Description
	NewSession	Open a new Vista window same as the current
	NewSessionAsk	Open a new Vista session, allowing a change in parameters.
	PrintScreen	Print the current Vista screen on the default printer
	Cut	Copy current selected text to the clipboard, and then delete it from the screen
	Copy	Copy current selected text to the clipboard
	Paste	Paste the contents of the clipboard on the screen at the current cursor location
	Reconnect	Disconnect and Reconnect the current session
	ReconnectAsk	Ask to reconnect the current session
	Undo	Resets the screen content back to what it was at the last host transmission
	User1.mac	Executes macro USER1.MAC, which is empty as supplied by Vista
	User2.mac	Executes macro USER2.MAC, which is empty as supplied by Vista
	User3.mac	Executes macro USER3.MAC, which is empty as supplied by Vista
	Record	Record keystrokes into a macro file
	Play	Play a pre-recorded or edited macro file
	Pause	Pause macro recording or playing for a moment
	Stop	Stop recording or playing a macro
	SendFile	Send a PC file to the host
	ReceiveFile	Receive a file from the host to the PC

	EditProfile	Edit the current Vista profile options
	EditKeyboard	Edit a keyboard map or mouse keys
	EditToolbar	Edit a toolbar
	ShowKeypad	Displays the selected keypad
	Clipboard	Run the Windows clipboard if available - clipbrd.exe
	Note Pad	Runs notepad.exe
	Session A	Runs a macro which will swap to session A, if that session is running
	Session B	Runs a macro which will swap to session B, if that session is running
	Session C	Runs a macro which will swap to session C, if that session is running
	HelpContents	Displays the Help Contents window

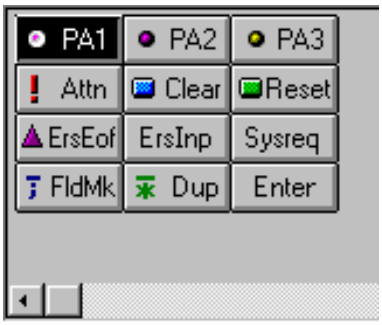
Keypad Editor

 The Keypad is normally hidden, and can be brought up by using the ShowKeypad function, which by default is assigned to the toolbar button shown at left.

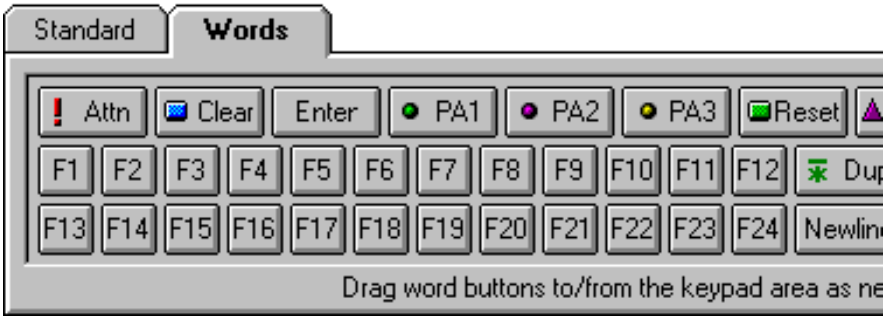
If you just want to find a keypad function, please see Keypad Defaults for a list of Vista default keypad settings. Otherwise, follow the instructions below for editing your own keypad layout.

How to Edit a Keypad Button

After starting the **Keypad Edit** dialog (from the **Options** menu), either select the button you want to modify in the Keypad Edit Area, such as shown below:



... or drag a new button from the Words button area to the Keypad Edit Area before modification, as shown here:



Then edit the Keypad function as you would any key on the keyboard. See Keyboard Edit if you don't already know how to do this.

Also, select the **Standard** tab to select buttons with graphical descriptions.

Keypad Edits

Adding Buttons

As shown above, simply grab a button from the Standard or Words area and drag it to the Keypad Edit Area in the desired location. Scroll the area vertically or horizontally as needed.

Moving Buttons

Simply grab the button in the Keypad Edit Area and move it to the position you want. When moving it a long distance, you may need to drop it, scroll the edit area, and then continue moving it. Buttons cannot be moved on top of other buttons.

Removing Buttons

To remove a button from the Keypad Edit Area, grab the button and drag it back to the Standard or Words area. It will then be removed from the keypad, leaving a hole.

Restrictions

The Repeat option is not allowed on Keypad buttons, so it is not shown.

Keypad Defaults

These are the default keypad buttons as provided by Vista at installation time. Of course you can change them to anything you want. See the Keypad Editor for information on how to do this.

Icon	Function	Description
	PA1	Sends a PA1 interrupt to the host.
	PA2	Sends a PA2 interrupt to the host
	PA3	Sends a PA3 interrupt to the host
	Attention	Sends an attention interrupt to the host
	Clear	Clears the screen and interrupts the host
	Reset	Resets the terminal emulator
	Erase End of Field	Erases text from the current cursor position to the end of the current input field
	Erase Input	Erases input from all fields on the current screen
	SysRequest	Sends a SysRequest interrupt to the host
	Field Mark	Types a field mark character on the screen
	Dup	Types a Dup character on the screen and repositions the cursor at the beginning of that field.
	Enter	Sends an Enter key interrupt to the host

File Transfer

Vista supports IND$FILE transfer. This is not very efficient for medium to large files, but works well for small files. Vista also supports multiple file transfers, for example, transferring an entire PDS back and forth by specifying dataset or member names with an asterisk wildcard character.

IND$FILE uses your TSO or CMS address space as a server, and the current Vista window as a client. This means (as you probably know) that your userid and the Vista session are both unavailable for other work during a file transfer.

If you have problems using IND$FILE transfer, then by all means get an FTP client going such as WS_FTP or many other available ones.

How to Transfer a File

First, make sure you are at TSO READY or CMS command prompt.

Select Transfer from the menu bar, and then select Send or Receive. "Send" in Vista means you are sending data from your PC to the host, and "Receive" means you are receiving data from the host to your PC.

Select or fill in the PC filename involved in the transfer. For the host dataset name, you can either type it in or try using the List and Member buttons (TSO only) to see what's currently available on the host.

Select other options such as the Text or Binary buttons, and make sure you indicate the host program you are running (TSO or CMS). Then press the Send or Receive button to initiate the transfer.

Press the View Log button to view the results if needed.

Note: File transfers can also be initiated from a macro. See SendFile and ReceiveFile for more information on this.

File Transfer Details

Transfer window items are the similar for both upload and download transfers, and consist of:

PC File

Specify the PC file name you want to send or receive, either by typing the name in, selecting from the current directory list, or by pressing the browse button to look in other directories. An asterisk wildcard character can be used for multiple file transfers.

Host File

Specify the Host file name you want to send or receive, either by typing the name in, selecting from the current directory list, or by pressing the browse button to look in other directories. An asterisk wildcard character can be used for multiple file transfers.

List

Press this button to read a list of filenames from the current host session into the Host File selection window, a screen at a time. The button changes to **More** when this function is working. Press the More key to continue gathering dataset names from the host, or select one for transfer.

Members

After selecting a dataset in the list previously added by the **List** button, use the **Members** button to list each member in the dataset. As with List, this button changes to **More** while busy. Press the More key to continue gathering member names from the host, or select one for transfer.

Auto

If you have a fast connection and don't want to press the **More** button when creating dataset or member lists, check this option and the **More** button will automatically be pressed for you.

List Level

In TSO only, the high-level-index typed here will be used when the **List** button is pressed. If this field is non-blank, single ticks will surround the dataset names in the resulting list. Otherwise, your default (profile prefix) high-level will be used.

Send or Receive

Press the Send or Receive button (depending on if you selected the send or receive dialog) to initiate the transfer. Status of the transfer is shown in the lower area of the transfer window. You can also open the log window for a history list.

Exit

Press to exit the transfer dialog. If a transfer is currently running, you will be asked if it's ok to cancel the current processing.

Options

Displays a screen where you can modify the program used for the IND$FILE style transfer, add extra parameters to the command, and other items. See Miscellaneous Options for more information.

Data

Select the style of data you are sending, and whether you want it appended to the end of an existing file or not. Typically, use **Text** mode for readable files such as payroll.txt, or jcl.cntl(idcams). Use **Binary** mode for just about everything else.

Host

Indicate what operating system you are using on the host - **TSO** (MVS) or **CMS** (VM)

Multiple File Transfer

The transfer windows support the transfer of multiple files to and from the host. If multiple files are selected, a macro file named VST$XFER.MAC is generated and run based on user input, which contains instructions for the multiple transfers.

Sending Multiple Files to the Host (Upload)

Select the Transfer/Send menu item, and use one of the following methods for sending multiple files to the host. In all upload cases, the files to be sent must appear in the Source List (PC).

- Type a source PC file name with a wildcard (asterisk) character, such as ***.txt**, and a target host name with a wildcard, such as **TEST.*.ASM** . The source filename will be used to replace the wildcard in the host name. If the host name is a PDS, such as **TEST.CNTL(*)**, you'll need to pre-define the PDS on the host prior to the upload.

- Select multiple files in the source PC file list with the mouse by holding the **shift and/or ctrl** key while selecting. If the target host name is blank, or if it does not contain a wildcard character, the original source filename will be used as the target name. If the target name contains an asterisk, then the source filename will be substituted for that wildcard.

Receiving Multiple Files from the Host (Download)

Select the Transfer/Receive menu item, and use one of the following methods for receiving multiple files to the host. In all download cases, you must have previously executed a LIST and/or MEMBERS function to obtain some items in the Source List (Host).

- Type a source host dataset name with a wildcard (asterisk) character, such as **TEST.*.ASM** or **TEST.PDS(*)**, and a target PC file name with a wildcard, such as ***.txt**. In this case, the wildcard-matching level of the source dataset name will be used to replace the wildcard in the PC file name when sending.

- Select multiple files in the Source dataset list with the mouse by holding the **shift and/or ctrl** key while selecting. In this case, the source host name will be used as the target PC filename, regardless of whatever is typed in the target PC name. Sorry for this restriction, I just can't think of a good way to determine which source datasetname level should be used for substitution.

Macros

Macros are Vista's method of doing repetitive or programmatic work with the emulator. The user can use the Record facility to record simple operations, or use the macro programming language to do more complex procedures.

Some notes:

- The Vista macro language is kind of a cross between the languages Basic and C. This wasn't really done on purpose - things just turned out that way as the macro processing code was developed.
- Vista macros are designed for small jobs. There are a limited number of variables and other items that will prevent any kind of large-scale program from running. Also, certain items you would expect in any language may be missing, such as support for variable arrays.
- Sometimes the word "parm" is used in this documentation in place of "parameter" or "param". Mainframe people are used to this.

Macro Recording and Playing

Macro Recording

To record a macro for a repetitive or difficult set of keystrokes, press the Record button, Macro/Record menu item, or right-click on a toolbar button previously defined to a macro.

Before recording, Vista will ask for the destination file name of the macro. You can use a name which is already pointed to by an existing ToolBar button, such as user1.mac, user2.mac or user3.mac, which have (by default) buttons on the toolbar. Or you can use a new name and execute it later by various methods.

You can also right-click any macro button on the toolbar (such as the 3 described above) and select the **Record Macro** option, without having to specify the actual macro file name.

A blinking "R" on the status bar indicates that keypresses are being recorded. Toolbar button hits and most menu item actions are also recorded. When you are finished recording, press the Stop key (or Macro/Stop menu item), and the macro will be stored.

Once defined, a new macro can be defined to a key or toolbar button using the Keyboard Edit or ToolBar Edit dialog.

Macro Playing

A Macro can be played (run) by assigning it to a key or toolbar button. A Macro also can be played by pressing the Play button, or by using the Macro/Play menu option.
In the latter case, you will need to specify the name of the macro each time it is run.

Macro Pausing

A Macro playing macro can be paused, which will stop it's action and still not exit the macro, allowing you to continue running the macro from where it was paused. Pressing a key or toolbar button while a macro is running will automatically pause it.

Press the Pause toolbar button, or use the Macro/Pause menu option to pause a running macro. Issue Pause again (or Play) to continue the macro from where it left off.

Macro Stopping

To stop a running macro issue the Stop function either by key, toolbar, or menu item. Macro's also have the ability to stop themselves by issuing the Stop function.

If a macro gets into a loop and causes Vista to hang up, you can usually stop the macro by pressing the **Pause** or **Break** key on the keyboard.

Notes: In some cases, a recorded macro may not play correctly. For example, by default macros will wait 20 seconds for the host to respond to input, and will fail if that time value is exceeded. In cases where a long host wait is normal, the user will have to increase the time-out value of the Wait function in the resulting macro text.

In other cases, the host will unlock the screen more than once before the transmission is complete, confusing Vista as to the time it should begin sending the next screen input text. Vista does it's best in these cases to wait for not only the screen to unlock, but also for a particular cursor position to be established. This works in most cases, but may sometimes require some modifications to the macro text to run properly, such as the manual addition of some **Wait(1)** statements.

General Macro Statement Format

- Macros are standard text format files, and must reside as *.MAC files in the Vista directory. Currently macros must be short (8 character) file names with a 3 character MAC extension.
- A macro line consists of a macro ‘statement’, such as a variable assignment statement, or a function statement.
- Each statement must fit on a single line of no more than 256 characters.
- Multiple statements on a line are possible if separated by a semi-colon character.
- A statement is considered a comment if the first non-blank character in the line is an asterisk. To add a comment at the end of a line, use ;*
- Blanks are normally ignored between fields in a statement.
- Numeric variables and numbers are coded as is. Strings must be coded between double quotes, such as "string". If a double quote is needed within a string, 2 double quotes must be coded, as in the string "this is ""quoted"" text"
- Upper and lower case text makes no difference when referencing functions or variable names, although code looks clearer if some upper/lower conventions are used. Also, when comparing strings or string variables with the “=“ operator, character case is ignored. Use the “==“ operator if case should not be ignored in the comparison.

Macro Variables

OverView

All Vista variables reside in a memory block which is created at the startup of an emulator session, and normally never cleared. This allows variables to exist between macro runs (and macro calls, when that facility becomes available). In programming terms, this means that all variables are global. Variables are not shared between Vista sessions, however.

Variables are not declared, but if you do attempt to use a variable that has not yet had some data assigned to it, you will get an error. System Variables, Constants, and Time Variables (through the use of the TimeDate function) are predefined by the system, and thus can be used without previous a assignment.

Variable Names

Variable names are 1 to 16 characters in length, must be alphanumeric, and must start with an alphabetic character. They can also contain (or begin with) the 4 "National Characters" I 'm sure all mainframe programmers are familiar with, namely, #, $, %, and @. Also, you can use the underscore character as an alpha character if you want.

When assigning or referencing a variable name, just code the name itself with no preceding ampersand or other character. In the Window Title and Status Text strings (in the Option dialog), you *must* code an ampersand in front of a variable name. In a macro, coding an ampersand prefix would be considered the 'and' logical operator and will most likely result in an error.

Variable Types

There are 2 types of variables, numeric and string. The actual variable type can change, depending on what type of data was last put into it, for example:

```
var1 = 3    ;* this makes var1 numeric
var1 = "3"  ;* now var1 becomes a string
```

When referring to a variable, however, the type must be appropriate for the function, otherwise an error will occur.

```
var1 = "3"    ;* var1 is now a string
a = 2 + var1  ;* this results in an error
```

Numeric variables are signed integers from -99999999 to 99999999
String variables can be from zero to 255 bytes in length

Macro Operators

Operators Types

Vista macro operators generally come in 3 styles, unary, binary, and logical:

Unary Operators

Unary operators operate on a single data element

- **+** Unary Plus has no effect on a data element, but is included for consistency
- **-** Unary Minus will negate the data element, for example, `a = -b`
- **!** Unary Not will logically complement the data element, for example, `a = !b`

Binary Operators

- **+** Addition and String Concatenation
- **-** Subtraction
- ***** Multiplication
- **-** Division
- **//** Remainder

Logical Operators

- **=** Equal - (uppercase/lowercase is ignored)
- **==** Explicit equal (uppercase/lowercase must match exactly)
- **!=** Not equal
- **<** Less than
- **>** Greater than
- **<=** Less than or equal to
- **>=** Greater than or equal to
- **&** Logical AND
- **|** Logical OR
- **^** Logical XOR
- **not** Logical NOT

Logical Operations

A logical operation operates on data that is normally considered to be a numeric 1, meaning true, or a numeric 0, meaning false. Logical operations always return either a 1 or a zero.

Logical numeric operations take into consideration the sign of the data involved. Logical text operations ignore case differences when comparing, i.e., the expression "ABCD" = "AbCd" returns a true value (use the “==“ operator if an explicit case-sensitive compare is necessary).

Operator Precedence

Operator precedence (the order in which operations are performed) follows the generally accepted standards described below. If 2 or more operators of the same level are seen, the operations proceed from left to right.

Parenthesis can change the order of the operations, if required. Also, redundant parentheses are allowed.

Highest to Lowest

Unary operators +, -, !
Arithmetic operators *, /, // (remainder)
Arithmetic operators + , -
Logical operations =, ==, !=, <, >, <=, >=
Logical operations &, |, ^, not

Macro Control Statements

The following statements are used to control the logic flow within a Vista macro:

Program Flow Control

If / Else / Endif
While / EndWhile
Select / Case / Default / EndSelect
For / Next
Break
Continue
Exit
Goto
Call / Subroutine / Return
Run

Note: With these statements, it's possible to put a macro into a never-ending loop. If you find your macro gets into such a loop, try the following:

- If an EditBox continues to appear within the loop, just press the "X" (close) button at the top right of the EditBox window. This causes macro processing to end.
- Try pressing the Pause key. This key is checked before each macro instruction is executed, and should (hopefully) cause the macro to end.
- If both attempts above fail, your only alternative is to use the ctrl-alt-delete keys, which calls up the Windows task list. You should be able to kill the task from that list, but that unfortunately ends the entire Vista session, not just the macro execution.

Macro Functions

Every statement other than the flow control items are functions, whether they are used within an expression, or used alone. Currently supported functions are:

Control Functions

Wait	Wait for various things to occur
ShortWait	Wait for times less than 1 second
Session	Switch to another Vista session
WinExec	Start another Windows program by executable filename
ShellExecute	Start another Windows program by data filename

Screen Functions

Type	Type a string of characters on the screen
Key	Issue a key function, such as Enter or Newline
Screen	Retrieve text from the current screen
Attribute	Retrieve attribute byte from the screen
Color	Retrieve color byte from the screen
Hilite	Retrieve hilite byte from the screen
OnScreen	See if text is anywhere on the screen

String Functions

Len	Return the length of a string
Mid	Return a substring of characters
Left	Return a substring starting at the left
Right	Return a substring starting at the right
Instr	Find a string within another string
Trim	Trim leading and trailing blanks from a string
Uppercase	Convert a string to upper case characters
Lowercase	Convert a string to lower case characters

Conversion Functions

Str	Convert a numeric variable to a string
Hex	Return the hex representation of a number
Asc	Convert a single character to it's ASCII numeric value
Chr	Convert a number to a single character
Val	Convert a numeric string to a number
TimeDate	Return formatted time and/or date string

File Functions

SendFile	Send a file from PC to Host via IND$FILE transfer
ReceiveFile	Receive a file from Host to PC via IND$FILE transfer
Open	Open a file or the clipboard for Get or Put
Get	Get a record from an open file or clipboard
Put	Put a record in an open file or clipboard
Close	Close a file or clipboard
Eof	Check for end of file when reading

Support Functions

Control	Control trace and other macro logic
Debug	Write string or number to Debug window

DebugWindow	Control the Debug window
MessageBox	Show a string of text in a Window's MessageBox
EditBox	Show a window with a data entry field to allow operator input
PutINI	Put a variable into the VISTA.INI file.
GetINI	Get a variable from the VISTA.INI file
SetOption	Set an option value
GetOption	Get an option value
PlaySound	Play a *.WAV file to create a sound
Defined	Determine if a variable is defined or not

If / Else / Endif Macro Functions

If ***expression***
... statements to perform if *expression* is true
Else
... statements to perform if *expression* is false
Endif

expression is any boolean expression

At least **If** and **Endif** statements must be coded. **Else** (and else statements) are optional.

Some examples:

```
a = 11
b = 30

If a = 10
  MessageBox("A is 10")
Else
  MessageBox("A is not 10")
EndIf

If b = 15 * 2
  MessageBox("B is 30")
EndIf
```

Note: Currently Vista macros do not allow the true-path or false-path statements to be executed to be on the same lines as the **If** or **Else** statements. For example, the following lines are **invalid**:

```
If (b = 15 * 2) MessageBox("B is 30")
Else MessageBox("B is not 30")
```

Note: Logical numeric operations take into consideration the sign of the data involved. Logical text operations ignore case differences when comparing, i.e., the expression "ABCD" = "AbCd" returns a true value (use the “==“ operator if an explicit case-sensitive compare is necessary).

While / EndWhile Macro Functions

While ***expression***
... statements to perform if *expression* is true
EndWhile

expression is any boolean expression

For example:

```
a = 1
While a < 5
  MessageBox("A is currently "+str(a))
  a = a + 1
EndWhile
```

Note: Currently Vista macros do not allow the true-path statements to be executed to be on the same line as the **While** statement. For example, the following line is **invalid**:

```
While (a < 5) a = a + 1
```

Select / Case / Default / EndSelect Macro Functions

Select ***expression1***
  Case ***expression2 [,expression3,expression4 ...]***
    ... statements to perform if *expression1 = any expression*
  Default
    ... statements to perform no *Case* statement matches
EndSelect

expression1 is any arithmetic, boolean, or string expression
expression2 (or the others) is any arithmetic, boolean, or string expression

As many **Case** statements as needed are allowed (well... until the internal scan table overflows)

An example:

```
a = 3
Select a
  Case 1
    MessageBox("a is 1")
  Case 2
    MessageBox("a is 2")
  Case 3
    MessageBox("a is 3")
  Default
    MessageBox("a is not 1, 2, or 3")
EndSelect
```

Note: if more than one **Case** *expression2* matches the **Select** *expression1*, only the first matching **Case** statements are executed.

Multiple expressions are allowed on a single **Case** statement, if separated by commas. For example:

```
a = 1
While a
  a = val(EditBox("Enter a Number from 1 to 20, or zero to stop"))
  Select a
    Case 10,11,12
      MessageBox("You typed 10 or 11 or 12")
    Default
      MessageBox("You typed something other than 10, 11, or 12")
  EndSelect
EndWhile
```

For / Next Macro Functions

For ***variable*** **=** ***start*** **to** ***end*** **step** ***increment***
... statements to perform for each iteration of loop
Next ***variable***

variable is any valid variable name
start is a numeric expression or constant start count
end is a numeric expression or constant end count
increment is a numeric expression or constant increment value

Some examples:

```
For row = 1 to 10
  For col = 1 to 10
    Debug(str(row)+","+str(col))
  Next col
Next row

For i = 5 to 100 step 10
  Debug(i)
Next i

For i = 10 to 1
  Debug(i)
Next i
```

Note: If the start count is larger than the end count, the default step increment will be set to -1, and the for next loop will automatically count backwards.

Break* *Macro Function

Break

No parameters

Break is used to break out of a **For/Next** or **While/EndWhile** loop.

Some Examples:

```
For a = 1 to 10
  If a = 5
    Break
  Endif
  Debug(a)
Next a

a = 0
While a < 10
  a = a + 1
  If a = 5
    Break
  Endif
  Debug(a)
EndWhile

For row = 1 to 10
  For column = 1 to 10
    If row = 5
      Break
    Endif
    Debug(str(row)+","+str(column))
  Next column
Next row
```

Note: Break exits only the most recent **For** or **While** loop. In the example above, only the column loop is exited, the row loop continues.

Continue Macro Function

Continue

No parameters

Continue is used to iterate a **For/Next** or **While/EndWhile** loop.

Some Examples:

```
For a = 1 to 10
  If a = 5
    Continue
  Endif
  Debug(a)
Next a

a = 0
While a < 10
  a = a + 1
  If a = 5
    Continue
  Endif
  Debug(a)
EndWhile

For row = 1 to 10
  For column = 1 to 10
    If row = 5
      Continue
    Endif
    Debug(str(row)+","+str(column))
  Next column
Next row
```

Note: Continue iterates only the most recent **For** or **While** loop. In the example above, only the column loop is iterated, the row loop continues.

Exit Macro Function

Exit nn

nn (optional) indicates a return code passed to an exit routine.

Exit causes the macro processing to stop immediately.

Example:

```
While (1)
  input = EditBox("Enter some data please, or type 'done'")
  if input = "done"
    MessageBox("I think you want to exit now, so I will")
    Exit
  endif
  MessageBox("You typed - " + input)
EndWhile
```

Goto Macro Function

Goto *label*

label is an identifier in the program to note a particular location

Example:

```
Loop:
  text = EditBox("Type a name")
  If text = "Tommy"
    Goto error1
  Else
    If text = "Thomas"
      Goto error2
    Else
      If text = "Tom"
        MessageBox("Very Good")
        Exit
      else
        MessageBox("Enter either Tommy, Thomas, or Tom please")
        Goto loop
      Endif
    Endif
  Endif

Error1:
  MessageBox("That sounds like a kid")
  Exit

Error2:
  MessageBox("That sounds too rich")
  Exit
```

Call / Subroutine / Return Macro Functions

Call *routine*
...
***routine* : Subroutine**
... statements to perform in subroutine
Return

routine is a label identifying the subroutine

Example:

```
a = 1
Call check

a = 2
Call check

a = 3
Call check

Exit

check: Subroutine
  Select a
    Case 1
      b = "a is equal to 1"
    Case 2
      b = "a is equal to 2"
    Default
      b = "a is not 1 or 2"
  EndSelect
  Call showresult
  Return

showresult: Subroutine
  Debug(b)
  Return
```

Notes:

Subroutines should be coded at the end of your macro.

All variables are considered “global” in Vista macros, so there are no variable parms passed either on the Call or Return statements. Just use global variables for parms, and be aware that any variables named within a subroutine are the same variables outside that subroutine.

Subroutines can call other subroutines, but a subroutine cannot be called recursively. For example, if we attempted to call the “check” routine from within the “showresult” routine above, that would be considered an error.

Run Macro Function

Run(*macroname*)

macroname is an 8 character macro name constant or variable

Example:

```
Run("macro2")
Run("macro3")
```

Notes:

Wait *Macro Function*

Wait(*nn*)
Wait(*nn*,*expression*)

nn - the number of seconds to wait
expression - boolean expression, if true, the wait is canceled

Waits for the *expression* to be true, or the number of seconds to expire.

If only the number of seconds is given, Wait will wait at that point in the macro for the given number of seconds.

If an expression is also given, the expression is evaluated at various times to see if the conditions are met. If met, macro processing continues. If not met within nn seconds, the macro fails with an error code.

Returns - nothing

Examples:

```
Wait(20)                      ;* Wait 20 seconds, then continue
Wait(10,Status="Unlocked")    ;* Wait for screen to unlock
```

ShortWait Macro Function

ShortWait(*nnnn*)

nnnn - the time to wait, in hundredth's of a second.

Waits for the timer to expire before macro processing continues.

Returns - nothing

Examples:

```
ShortWait(10)                ;* Wait 1/10 of a second
ShortWait(150)               ;* Wait a second and a 1/2
```

Session Macro Function

Session(*text*)

text - a single character

Switches to the Vista session indicated by the given character. If the indicated session is not active, there is no error returned.

Returns - nothing

Examples:

```
Session("C")
Session("A")
```

WinExec Macro Function

WinExec(*text*)

text - a Windows executable filename

Enables Vista to startup other Windows functions, such as Notepad.exe or Clipbrd.exe, from a macro.

Returns - nothing. The macro does not wait for the executed function to complete.

Examples:

```
WinExec("notepad.exe vista.txt")
WinExec("clipbrd.exe")
```

Also see ShellExecute if you want to let Windows determine the registered executable for a particular data file.

ShellExecute Macro Function

ShellExecute(*operation,file,parms,dir,showcmd)*

operation - must be “open” or “print”
file - the executable or data file to perform the operation on
parms - additional parameters for the executable
dir - the default directory
showcmd - indicates how to show window when opened

ShellExecute differs from Winexec in that you can pass it the name of a data file, and let Windows figure out what program should be used to open, print, or explore that particular item.

Returns a number which is an “instance” handle windows uses to locate the open window.

The last parm, *showcmd*, is a number indicating how the window should be opened, and can be:

1 - Shows the window normally
2 - Minimizes the window at startup
3 - Maximizes the window at startup

Examples:

```
ShellExecute("open","www.cbttape.org","","",1)
ShellExecute("print","vista.txt","","",1)
```

All 5 parms must be supplied, even if they are just null strings as shown above.

Type *Macro Function*

Type(*text*)

text - a string of characters

Types the string of text on the screen at the current cursor location, and updates the cursor location as necessary. Only text data can be included, keys such as Tab and Enter must be issued with the **Key** macro function.

Returns - nothing

Examples:

```
Type("Logon")
Key("Enter")
Type("PDF 2")
Key("Enter")
```

Key Macro Function

Key(*function*)

function - string name of a Vista function

Issues a single keypress for non-character emulator Functions

Returns - nothing

Examples:

```
Key("Tab")
Key("Enter")
Key("SendFile")
```

Most functions are specified as a single parameter, as shown above. The following functions, however, allow optional secondary parameters:

MoveCursor, MoveCursorEnter,
SelectLine, SelectField, SelectText, SelectWord, SelectJCL,
SelectLocation1, SelectLocation2

All of these use the current mouse location as the row/col specification for the function, unless row and col are supplied.
Some examples:

```
Key("MoveCursor")          ;* move cursor to mouse position
Key("MoveCursor",row,col)  ;* move to specific row and column
Key("SelectWord")          ;* select word at cursor position
Key("SelectWord",row,col)  ;* select at specific row and column
```

PrintFile

```
Key("PrintFile")           ;* prints pc$print.txt
Key("PrintFile",filename)  ;* prints the specified filename
```

Screen* *Macro Function

Screen(*row,col*)
Screen(*row,col,length*)

row - numeric row, from 1 to screen height
col - numeric column, from 1 to screen width
length - optional numeric length of string to return

Returns a string of characters at *row,col* on the current screen. If *length* is not supplied or causes the line to go beyond the right edge of screen, only that text up to the right edge is returned.

Examples:

```
Wait(10,Screen(2,20,7)="Welcome")

If Screen(10,20) = "V1234567"
  MessageBox("I see the string I'm looking for")
EndIf
```

Attribute Macro Function

Attribute(*row,col*)

row - numeric row, from 1 to screen height
col - numeric column, from 1 to screen width

Returns the attribute byte (0 to 255) located at the *row,col* location.

Example:

```
a = Attribute(10,20)
```

Color *Macro Function*

Color(*row,col*)

row - numeric row, from 1 to screen height
col - numeric column, from 1 to screen width

Returns the color byte (0 to 255) located at the *row,col* location. Note that the left 4 bits in the color byte indicate the screen background color, and the right 4 bits indicate the text or foreground color. Colors may not be standard, since they can be changed with the Options dialog.

Example:

```
a = Color(10,20)
```

Hilite *Macro Function*

Hilite(*row,col*)

row - numeric row, from 1 to screen height
col - numeric column, from 1 to screen width

Returns the hilite byte (0 to 255) located at the *row,col* location.

Example:

```
a = Hilite(10,20)
```

OnScreen *Macro Function*

OnScreen(*text*)
OnScreen(*text*,*row*)
OnScreen(*text*,*row*,*col*)

text - string of text we are looking for
row - numeric row, from 1 to screen height
col - numeric column, from 1 to screen width

If *text* only is supplied, the **OnScreen** function returns TRUE if the string of *text* can be found anywhere on the screen.

If *row* only is supplied, then only that *row* is searched for *text*

If *row,col* are both supplied, then the *text* must match exactly at that location in order to evaluate TRUE.

Examples:

```
Wait(10,OnScreen("Welcome",2))

If OnScreen("Welcome")
  MessageBox("The word ""Welcome"" is on the screen")
EndIf
```

Len Macro Function

Len(*text*)

text - a string of characters

Returns the numeric length of string *text,* which includes any leading or trailing spaces. Returns zero for an empty string.

Example:

```
a = Len("Hello there")        ;* result is 11
```

***Mid* Macro Function**

Mid(*text,start,length*)

text - a string of characters
start - starting location within string
length - length of string to return

Returns a substring of *text* based on *start* and *length*.
If *length* causes the string to go past the length of text, the returned string may be shorter than *length.* If *start* is not within *text,* the returned string is empty.

Examples:

```
a = "ABCDEFGHIJKL"
b = Mid(a,3,5)                ;* result is "CDEFG"
c = Mid(Mid(a,5,10),3,3)      ;* result is "GHI"
```

Left *Macro Function*

Left(*text*,*length*)

text - a string of characters
length - length of string to return

Returns a substring of *text* starting at the left and continuing for *length* characters, or until the end of the string if *length* is longer than *text.*

Examples:

```
a = "ABCDE"
b = Left(a,3)    ;* result is "ABC"
c = Left(a,10)   ;* result is "ABCDE"
```

Right* *Macro Function

Right(*text, length*)

text - a string of characters
length - length of string to return

Returns the right characters from *text* as determined by *length.* If *length* is longer than the string, then the entire string is returned.

Examples:

```
a = "ABCDEFG"
b = Right(a,3)          ;* result is "EFG"
c = Right(a,10)         ;* result is "ABCDEFG"
```

Instr Macro Function

Instr(*startloc, string1, string2*)

startloc - location within *string1* to start looking
string1 - the string to search
string2 - the string we are searching for

Searches *string1* for the occurance of *string2*

Returns the numeric character position of *string2*, if it was found in *string1* somewhere past *startloc*. If the string was not found, this function returns zero.

Examples:

```
a = Instr(0,"Try to find","to")      ;* result is 5
a = Instr(0,"Hello","ll")            ;* result is 3
a = Instr(8,"Try to find","to")      ;* result is 0
```

Trim *Macro Function*

Trim(*string*)

string - input string to trim

Returns the input string with leading and trailing blanks removed

Examples:

```
a = Trim("  Hello  ")         ;* result is "Hello"
```

Uppercase Macro Function

Uppercase(*string*)

string - input string to convert

Returns the input string to convert to upper case

Examples:

```
a = Uppercase("Hello 12mM")   ;* result is "HELLO 12MM"
```

Lowercase Macro Function

Lowercase(*string*)

string - input string to convert

Returns the input string to convert to lower case

Examples:

```
a = Lowercase("Hello 12mM")   ;* result is "hello 12mm"
```

Str Macro Function

Str(*number*)

number - a numeric value or variable

Returns the string representation of the supplied numeric value.

Examples:

```
a = Str(12345)            ;* result is "12345"
CursorPos = Str(row)+","+Str(col)
```

Hex Macro Function

Hex(*number*)

number - a numeric value or variable

Returns the hex string representation of the supplied numeric value.

Example:

```
h = Hex(23456)          ;* result is "00005BA0"
```

Asc Macro Function

Asc(*string*)

string - a single character

Returns the numeric ASCII representation of the supplied character.

Example:

```
n = Asc("A")     ;* result is 65
```

Note: The mainframe works with EBCDIC of course, but all characters on the PC (including what you see on the Vista screen) have been translated into ASCII.

Chr Macro Function

Chr(*number*)

number - a numeric value or variable

Returns the ASCII character representation of the supplied numeric value.

Example:

```
c = Chr(48)      ;* result is "0"
```

Note: The mainframe works with EBCDIC of course, but all characters on the PC (including what you see on the Vista screen) have been translated into ASCII.

Val Macro Function

Val(string)

string - a string of the characters 0 through 9

Returns the numeric decimal value of the string. The string of numeric characters to convert is considered to end either at the end of the string, or with the first non-numeric character seen.

Examples:

```
a = Val("23456")        ;* result is numeric 23456
a = Val("12ABC")        ;* result is numeric 12
a = Val("XYZ")          ;* result is numeric 0
```

TimeDate Macro Function

TimeDate(string)

string - format specification for the time/date string you want returned.
Formats are described in Time/Date Variables.

Returns a string

Examples:

```
td = TimeDate("%m/%d/%y")      ;* result  12/20/97
td = TimeDate("%Y.%j")         ;* result  1997.354
td = TimeDate("%H:%M")         ;* result  22:54
```

SendFile Macro Function

SendFile(*pcfile,hostfile,parm1,parm2,parm3*)

pcfile - string indicating the source file on the PC
hostfile - string indicating the target file on the host
parms - up to 3 parms can be specified, chosen from:

Text or **Binary**	transfer mode
Replace or **Append**	allocation mode
TSO, **CMS**	host application

Returns nothing at this time. You will need to check the transfer log for the results of the transfer (sorry).

Example:

```
SendFile("vista.txt","'testid1.my.data'","tso","text","replace")
```

- You should specify each of the parms for transfer mode, allocation type, and host application, otherwise the parameters used will be from the previous upload. Also, any parms used in a macro will replace those settings on the file transfer window.

- Surround parms with quotes, since they are treated as string constants (and in fact, could be variables).

- Include single ticks within the *hostfile* name if you want to indicate the high-level-index for TSO transfers.

- As with any file transfers, make sure your application is ready to accept the IND$FILE command before running the macro (for example, TSO READY mode).

ReceiveFile Macro Function

ReceiveFile(host*file*,*pcfile*,*parm1*,*parm2*,*parm3*)

hostfile - string indicating the source file on the host
pcfile - string indicating the target file on the PC
parms - up to 3 parms can be specified, chosen from:

Text or **Binary**	transfer mode
Replace or **Append**	allocation mode
TSO, **CMS**	host application

Returns nothing at this time. You will need to check the transfer log for the results of the transfer (sorry).

Example:

```
ReceiveFile("testid1.my.data","vista.txt","tso","text","replace")
```

- You should specify each of the parms for transfer mode, allocation type, and host application, otherwise the parameters used will be from the previous upload. Also, any parms used in a macro will replace those settings on the file transfer window.
- Surround parms with quotes, since they are treated as string constants (and in fact, could be variables).
- Include single ticks within the *hostfile* name if you want to indicate the high-level-index for TSO transfers.
- As with any file transfers, make sure your application is ready to accept the IND$FILE command before running the macro (for example, TSO READY mode).

Open *Macro Function*

Open(*filename,mode*)

filename - the string name of the file that we want to open, or
"clipboard" for clipboard access
mode - string of "input", "output", or "append"

Returns the numeric handle of the open file, for use in Get or Put operations.
If open fails, the reason code is returned.

Examples:

```
*-------- Write a single record to a file ---------

handle = Open("vista.txt","output")
Put(handle,"This is a record to write")
Close(handle)

*-------- Read all records in an existing file ----

handle = Open("c:\autoexec.bat","input")

While not(Eof(handle))
  textline = Get(handle)
  Debug(textline)
EndWhile

Close(handle)

*-------- Read records from the clipboard ---------

handle = Open("clipboard","input")

While not(Eof(handle))
  textline = Get(handle)
  Debug(textline)
EndWhile

Close(handle)
```

Get Macro Function

Get(*handle*)

handle - numeric handle of a file opened for input

Returns a string of data read from a file previously opened for input

Example:

```
*-------- Read all records in an existing file ----

handle = Open("c:\autoexec.bat","input")

While not(Eof(handle))
  textline = Get(handle)
  Debug(textline)
EndWhile

Close(handle)
```

Put Macro Function

Put(*handle,text*)

handle - numeric handle of a file opened for output
text - text to be written to file

Writes text string to a file previously opened for output. A carriage return and line feed are appended to the end of each record when writing.

Examples:

```
*-------- Write a records to a file ---------------

handle = Open("vista.txt","output")
For i = 1 to 10
  Put(handle,"This is a record number "+str(i))
Next i
Close(handle)

*----- Write a records to the clipboard --------

handle = Open("clipboard","output")
For i = 1 to 10
  Put(handle,"This is a record number "+str(i))
Next i
Close(handle)
```

Close* *Macro Function

Close(*handle*)

handle - numeric handle of an open file

Closes a file

Example:

```
Close(handle)
```

Note: If the file is not currently open, no error results

Eof Macro Function

Eof(*handle*)

handle - numeric handle of an open file

Returns TRUE (1) if the end of file has been reached on the currently open file, otherwise, this function returns FALSE (zero)

Example:

```
*-------- Read all records in an existing file ----

handle = Open("c:\autoexec.bat","input")

While not(Eof(handle))
  textline = Get(handle)
  Debug(textline)
EndWhile

Close(handle)
```

Control *Macro Function*

Control(*flags*)

flags - a number containing various bit flags

The bit flags tell the macro control logic to do certain things. Flags currently include:

0x0001 TraceMessage Trace each macro line to debug window

Example:

```
Control(TraceMessage)
```

Debug Macro Function

Debug(*variable*)

variable - a string or numeric variable

Write the variable to the Debug Window. If the debug window is not currently open, this function will cause it to be opened.

Examples:

```
Debug(12345)
Debug("12345")
```

Note: A keyboard function or toolbar button can be set to "DebugWindow" if desired (opens the window). Also, use the DebugWindow macro function to open, close, clear, and temporarily hide the Debug window as needed.

DebugWindow Macro Function

DebugWindow(*function*)

function - one of the following text strings:

open - opens or redisplays the debug window
close - closes the debug window (data is lost)
clear - clears the debug window (data is lost)
hide - hides the debug window (data is retained)

Examples:

```
DebugWindow("clear")
DebugWindow("hide")
```

MessageBox* *Macro Function

MessageBox(*message*,*title*,*icon*)

message - a string containing the message you want to show
title - (optional) string that appears on the title bar of the message box
icon - a number indicating the icon on the message box, including:

0 - no icon
1 - stop
2 - information
3 - exclamation

The message box always has an "Ok" button which must be pressed.
No return value is given from this function.

Examples:

```
MessageBox("Here is a message",3)
MessageBox("Another message","Title Line",2)
MessageBox("The last message")
```

EditBox Macro Function

EditBox(*message,title,passwordchar,length,width*)

message - a string containing the message you want to show
title - (optional) string that appears on the title bar of the message box
passwordchar - (optional) a character used to hide password entry
length - (optional) maximum length of the input string
width - (optional) width (in pixels) of the input area

This function causes a window to appear where the user can type a single line of text that is returned to the macro.

Most parameters are optional, yet positional. For example, to use a *passwordchar* string, you must code a *title* string, otherwise Vista will be confused. And to specify *width*, you must specify *length*.

Examples:

```
password = EditBox("Enter your password","Password","*",8)
name = EditBox("What is your name?",30)
```

PutINI Macro Function

PutINI(*varname*,*value*)

varname - the name of the variable you want to store
value - the data you want to store

This function allows you to store variables in the VISTA.INI file (in your windows directory).

Remember that the *varname* specified must be surrounded with quotes if you want to specify the variable name as a constant in this function.

Examples:

```
var4 = 123456
var5 = "This is a string of text"
PutINI("var3","This is the text placed in the INI file")
PutINI("var4",var4)
PutINI("var5",var5)

Debug(GetINI("var5"))    ;* check one of the variables
```

Note that any numeric variables (such as “var4” above) will be converted to a text string while saving in the INI file.

Variables are stored in the [UserVariables] section of the VISTA.INI file. There is no Vista macro function to delete them once they are added, but it’s easy to just edit the VISTA.INI file directly with any text editor.

GetINI Macro Function

GetINI(*varname*)

varname - the name of the variable you want to store

Returns the value of *varname* if that variable was found in the VISTA.INI file (in your windows directory). If the variable was not found, this function returns an empty string.

Variables are searched for in the [UserVariables] section of VISTA.INI. Other sections are not available to be obtained by this function.

Examples:

```
var4 = GetINI("var4")
var5 = GetINI("var5")
```

All variables are returned as strings.

SetOption *Macro Function*

SetOption(*tabname,optionname,value*)

tabname - the name of the tab on the options panel
optionname - the text of the option you want to change
value - the text or numeric value to assign to the option

Vista has many options on the Options/Options menu item. The SetOption function allows you to set most of them via a macro statement. Remember that any option you set will be eventually saved in the currently active session file.

Specify the *tabname* as the text on one of the "tabs" in the Options window, such as "General" or "Cursor". Specify the *optionname* as the text of the value you want to change, without any spaces, such as "WindowTitle" or "Row/ColonStatusLine". Character case is ignored, but can make the statement more readable.

Text Fields:

Text fields can be set by supplying the entire new text string, such as the following examples:

```
SetOption("General","WindowTitle","This is the new title")
SetOption("Print","UserName","Tom Brennan")
```

Check Boxes:

Check boxes are set or cleared by passing the numbers 1 or 0 (True or False) to check or uncheck the specified options, such as:

```
SetOption("General","AskBeforeClosing",True)
SetOption("Display","RowColOnStatusLine",False)
```

Pulldown Lists:

Pulldown Lists typically allow a text string to be assigned, but in most cases this string is not checked for validity before plugging it into the *.ses file. This could cause the session file to become unusable, so be careful.

```
SetOption("General","EditKeyboard","good.key")
```

Radio Buttons:

Radio Buttons are those round buttons that allow a one-of-many selection. For the SetOption function, you must supply the name of the box surrounding the buttons, rather than the name of the button itself. The value is a number (starting with zero) indicating which button to set, for example:

```
SetOption("Cursor","Ruler",3) ;* set the crosshairs active
SetOption("Display","TerminalModel",2)  ;* set to mod4
```

Not all options are changeable using the SetOption function. And some, such as changing the terminal model, require you to reconnect to the host.

GetOption Macro Function

GetOption(tabname,optionname)

tabname - the name of the tab on the options panel
optionname - the text of the option you want to change

The GetOption function allows you to query the value of most Vista options. The value returned may be a number or a string of text, depending on the selection option.

Specify the *tabname* as the text on one of the "tabs" in the Options window, such as "General" or "Cursor". Specify the *optionname* as the text of the value you want to change, without any spaces, such as "WindowTitle" or "Row/ColonStatusLine". Character case is ignored, but can make the statement more readable.

Text Fields:

Text fields return the value of the field, for example:

```
windowtitle = GetOption("General","WindowTitle")
myname = GetOption("Print","UserName")
```

Check Boxes:

Check boxes return either numeric 1 or 0 (True or False) to indicate whether the option is checked or unchecked, such as:

```
ask = GetOption("General","AskBeforeClosing")
rowcol = GetOption("Display","RowColOnStatusLine")
```

Pulldown Lists:

Pulldown Lists return the text value of the currently specified entry, such as:

```
keyfile = GetOption("General","EditKeyboard")
```

Radio Buttons:

Radio Buttons are those round buttons that allow a one-of-many selection. For the GetOption function, you must supply the name of the box surrounding the buttons, rather than the name of the button itself. The returned value is a number (starting with zero) indicating which button is currently set, for example:

```
ruler = GetOption("Cursor","Ruler")
model = GetOption("Display","TerminalModel")
```

Not all options are queryable using the GetOption function.

PlaySound Macro Function

PlaySound(soundfile)

soundfile - the filename of a WAV format sound file

The sound is played through the computer's sound card. Control returns immediately to the macro, even before the sound completes. If Vista is already playing a sound, that sound is interrupted. Only one sound can be playing at a time.

Example:

```
PlaySound("beep1.wav")
```

Defined Macro Function

Defined(*varname*)

varname - the variable we are checking

Returns TRUE if the variable is defined to Vista. Note that all variables are global and remain intact for the entire session, even across macro calls.

Example:

```
If Defined("var1")
  Debug("var1 has already been defined")
Else
  Debug("var1 has not yet been defined")
Endif
```

Variables

Variables can be used in macros, and also in text strings such as the Window Title and Status Line text areas. There are actually 3 types of variables:

System Variables	Vista variables, such as CursorPos and Status
Time/Date Variables	Time and date format variables, in C language format
Constants	Vista constants for general macro usage

System Variables

System variables can be referenced from macros as listed, and also from the Window Title and Status Line text strings in the General Options panel as long as they are prefixed with an ampersand (&) character. Variable names are shown here in upper and lower case for readability, but the names in actual use are not case-sensitive.

Some variables, such as CursorPos, can be changed within a macro to cause some effect on the current screen. Others are read-only and cannot be changed. Some variables return a string, and others return a numeric value.

Read-Only System Variables

EndCol	Column position of the lower right corner of the selection box. See SelectBoxActive variable for additional information.
EndRow	Row position of the lower right corner of the selection box. See SelectBoxActive variable for additional information.
FontSize	Current Font Size NNxNN
HostAddr	Host IP Address
HostName	Current host name. May be an Alias pointer to a real IPname (see below)
InputLen	Length of the last input buffer
IPname	Host Domain Name Server name, or numeric IP address
KeyScanCode	Hardware numeric code produced by the last key press
LocalAddr	Local IP Address
LocalName	Local Computer Name
LUname	VTAM LU name when TN3270E mode is active
PassedKey	This is a text representation of the last key typed when certain user exits are active. See User Exits for more information.
Port	Current IP Port number
ScreenHeight	Height of the current display area, in characters
ScreenWidth	Width of the current display area, in characters
SelectBoxActive	Set to True (1) if a selection box was drawn on the screen when the macro was started. If so, StartRow, EndRow, StartCol, and EndCol variables indicate the location of the selection box. If SelectBoxActive = False, no selection box was drawn on the screen when the macro started, and the row and col variables indicate the location of the previous selection box.
SessName	Current Vista session name
Socket	Current Socket number
StartCol	Column position of the upper left corner of the selection box. See SelectBoxActive variable for additional information.

StartRow Row position of the upper left corner of the selection box. See SelectBoxActive variable for additional information.

Status String variable indicating the current screen status, which may be one of the following values:

NOTCONNECTED	Host is not connected
START	Host connection is starting
SESSION	Host connection is in session dialog
CONNECTING	Host is attempting connection
CONNECTED	Host is connected
UNLOCKED	Screen is unlocked
LOCKED	Screen is locked, waiting for host
SYSLOCK	Screen is locked, waiting for host
PROTECTED	Attempting to type in protected field
INSERTOVERFLOW	Attempting to insert too many characters in field
TR_READY	File transfer ready to go
TR_DATA	File transfer data being recieved
TR_MESSAGE	File transfer message being received

TermModel Terminal model number

TNmode Indicates current telnet communication mode, either “TN3270” or “TN3270E”

TpxSwap For CA-TPX session manager users. If the TPX Netspy setting is active, TPX will send out an invisible data block whenever you swap sessions. Vista picks the session name out of this data and moves it to the TpxSwap variable, which can be displayed on the status bar, or used in Vista’s TPXSWAP exit.

TranTime Time of last transaction in the string format NNN.N

Version Vista current software version and release number

WindowID Current Window ID letter (A-Z)

Read-Write System Variables

CursorPos Position of cursor on the screen, in the string format of "row,col".

CursorRow Numeric row of the current screen cursor.

CursorCol Numeric column of the current screen cursor.

Time/Date Variables

Time and date variables can be referenced in the Window Title and Status Line fields on the General Options panel, as shown below. For referencing in a macro, however, you must use the TimeDate function. For time/date variables, character case is important and you must prefix the character with a percent sign. Also, if you intend to use a real percent sign in your string, you must use 2 percent signs together (%%).

%a	Abbreviated weekday text name
%A	Full weekday text name
%b	Abbreviated month text name
%B	Full month text name
%c	Date and Time representation
%d	Day of the month (01-31)
%H	Hour in 24-hour format (00-23)
%I	Hour in 12-hour format (01-12)
%j	Julian day of the year (001-366)
%m	Month number (01-12)
%M	Minutes (00-59)
%p	AM/PM indicator
%S	Seconds (00-59)
%U	Week of the year, where Sunday is the first day of the week
%w	Weekday as a number (Sunday =0 Saturday=6)
%W	Week of the year, where Monday is the first day of the week
%x	Date
%X	Time
%y	2 digit year
%Y	4 digit year
%z	Time zone string

For example, the string `%m/%d/%y.%j %H:%M` is converted to `12/20/97.354 22:54`

Constants

Constants are simply read-only variables that contain a useful constant for macro processing:

True	numeric 1 - can be used in boolean expressions
False	numeric 0 - can be used in boolean expressions

Controlling Vista

User Exits

User exits are a way of adding your own code (in the form of Vista macros) to various spots in Vista processing.

Host Control

The Host can pass macro statements to Vista to be executed, by running the VSTEXEC command under TSO.

User Exits

User Exits are various points in Vista processing where you can place your own code for the purpose of tailoring the product to better meet your needs.

User exits are files named *.ext, and are text files in the Vista macro language. They are named with a different extension so they won't show up when listing macros for regular use.

To enable an exit, simply include the proper file in your Vista directory. At startup, the program checks for their existence, and if found, calls them at various points.

PROTECT.EXT

This exit gets called each time the user attempts to type on top of a protected field. To enable this exit, create a macro file named PROTECT.EXT using any text editor, and place it in the Vista directory.

Typically, you might use this to cause the cursor to move to an input field if you attempt to type on top of a protected field. For example, Key("Home") in this macro will cause the cursor to be moved to the home field. Another option might be Key("Tab") to automatically move the cursor to the next field on the screen.

Or you could code more elaborate processing, for example, to determine if the command line happens to be on the bottom or top of the screen, and move the cursor appropriately.

TPXSWAP.EXT

If you are running a program from Computer Associates called TPX, you've probably often wondered which session you are currently viewing (for example, before issuing a critical command). The typical way to be sure is to back out to the TPX main menu, and re-select the session you are interested in, even though you may have already been there.

I've always wanted a way to display my current TPX session somewhere on the screen. But since you can't interfere with the actual screen data, the place for this is ***outside*** the screen area, for example, on the emulator title or status bar.

But how can the terminal emulator know what TPX session you are currently viewing? And how can the emulator know when a session switch has occurred? Well, another CA product, Netspy, also needs to know this same information. To solve this, the developers have a "Netspy" option within the TPX setup parameters which, when enabled, will cause TPX to send the session name out to the terminal (at session swap time) in invisible text. You can't see this information, but Netspy - and a terminal emulator - can see it. This option can be enabled for TPX regardless of if you have Netspy or not.

When you have a Vista TPXSWAP exit active, by creating a Vista macro called TPXSWAP.EXT and restarting Vista, this exit gets control whenever one of these invisible TPC session switch messages is received. Vista grabs the TPX session id from the message, places it in a variable named TPXSWAP, and then calls the exit.

in the exit, of course, you can do what you want with the TPXSWAP variable, such as adding it to the title or status bar.
The only downside I've found is that once you've gotten your TPX administrator to enable the Netspy function, you might notice a short delay while swapping sessions due to the extra data coming from the host at that time.

LOCKED.EXT

This exit gets called when a user attempts to type on a locked keyboard. Its purpose is to react to typing on a terminal that is continuously locked, such as an SDSF screen in AUTO UPDATE mode. For example, the following macro would issue an Attention to unlock the keyboard and allow typing, when a user attempts to type on the locked screen:

```
************************************************
* locked.ext exit
************************************************

If Screen(2,21,16) = "**** AUTO UPDATE"
  Key("Attention")
  Key("ResetKeyBuffer")
  Wait(20,status="unlocked")
  function = Right(passedkey,Len(passedkey)-2)
  Select Left(passedkey,1)
    Case "C"
      Type(function)
    Case "A","E","O"
      Key(function)
    Case "M"
      Run(function)
  EndSelect
  Exit(8)
EndIf
```

Upon input, this exit is passed the "passedkey" variable, which is in the format of:

x-function

where: x = C, A, E, O, or M indicating the function is a Character, Action, Edit, Operation, or Macro. This will be needed in order to execute the passed function properly.

function = the text representation of the function executed while the screen was locked.

examples: C-a the character "a" was typed
A-Enter the enter key was pressed

Upon exit, you should provide a return code using the Exit statement, as shown in the example code above.

Exit(0) - default, causes Vista to continue normally
Exit(8) - causes Vista to interrupt normal key processing

VSTEXEC Host Control

The TSO program **VSTEXEC** can be used to control the emulator from the Host. The TSO program sends special instructions to the emulator, and data in the form of Vista macro statements. These macro statements are then executed on the PC side.

For example, this command can be used to initiate a file transfer from the host, by issuing the following from TSO READY mode:

```
VSTEXEC SendFile("c:\source.dat","target.data","tso","replace","text")
```

And like any macro, you can code multiple statements on a single line, such as:

```
VSTEXEC a=10;b=20;c=a+b;Debug(c)
```

For more complex macros, you can specify a datasetname or ddname on the VSTEXEC command, pointing to an LRECL=80 RECFM=FB dataset. Records are read and passed to Vista, and then executed when the transfer is complete. For example:

```
VSTEXEC DSNAME(TOM.MACROS(TEST1))
VSTEXEC DDNAME(MACRO)
```

See comments in the assembler source file for more information, such as error codes, and calling conventions when called from another program. Source code is available in the VSTEXEC.ASM file included with this installation. You'll need to assemble the program on your OS/390 system and then place the resulting load module in a spot (linklist, etc.) where it can be executed.

Macro Error Messages

Vista produces various error messages, as described here:

Macro errors usually result in a message window containing the line number where the error occurred and a short description of the error:

Code	*Description*
0	Unknown Error - Contact Vista Support
1	Keyword or Variable name is too long
2	Numeric constant is too many characters
3	Numeric constant is too large
4	Special character string is too long
5	If without Endif was seen
6	Unknown keyword seen
7	Unknown parameter seen
8	Unexpected text seen
9	While without EndWhile was seen
10	Variable or function not found
11	Unknown operator
12	Unbalanced parenthesis
13	Incompatible variable type for operation
14	Operation invalid for text variable types
15	Unknown function
16	Invalid number of parms for function
17	Invalid type of parm for function
18	Operand seen where an operation was expected
19	No more available file handles
20	Invalid open-file action
21	File open error
22	File is not open
23	Record too long
24	Invalid file handle
25	End of file
26	Left parenthesis expected to follow function name
27	Invalid DebugWindow parm
28	Variable name is too long
29	Reserved system variable cannot be altered
30	Invalid Key Function
31	Timeout while waiting for condition to be true
32	Variable table overflow
33	Cursor move key functions need row,col parms
34	NextSession parm invalid
35	Else was seen without a previous If
36	EndIf was seen without a previous If
37	Internal error - Invalid scan table address
38	EndWhile was seen without a previous While
39	Case was seen without a previous Select
40	Default was seen without a previous Select
41	EndSelect was seen without a previous Select
42	Next was seen without a previous For
43	Assignment not seen on a For statement
44	For statement was seen without To parameter
45	For statement must be all numeric values
46	Step is the only statement that can follow To

47 Value for Step cannot be zero
48 Next variable does not match previous For
49 String variable too long (more than 255 bytes)
50 Break invalid outside of For/While/Select block
51 Continue invalid outside of For/While block
52 Scan area overflow. Program may be too large
53 Goto can only have a label
54 Label not found
55 File Transfer parm error. See Help info for an example
56 Variable name is a reserved word or function name
57 An error occurred while writing to a file
58 Get function cannot be used for file open for output
59 Put function cannot be used for file open for input
60 Clipboard is full
61 Invalid Parameter Value
62 Call function does not specify a subroutine address
63 Goto function does not specify a label address
64 Return seen outside of a subroutine
65 Attempt to call a subroutine recursively

Macro Error Code 0

Unknown Error - Contact Vista Support

Obviously this shouldn't happen. Please send me an email, with the failing macro as an attachment if possible - especially if you can reproduce this error every time.

Macro Error Code 1

Keyword or Variable name too long

Keyword, function, and variable names must be 16 characters or less, must be alphanumeric only, and must begin with an alpha character or the national characters @#$%

Macro Error Code 2

Numeric constant is too many characters

When specifying a numeric constant, such as in the code:

```
a = 12345
```

... the number was longer than 8 characters. Numbers in Vista macros must be integers from -99999999 to +99999999

Macro Error Code 3

Numeric overflow during evaluation

When evaluating a numeric expression, the result of an operation produced a result that was less than -99999999 or greater than +99999999, such as this example:

```
a = 12345678 * 10 / 100
```

Macro Error Code 4

Special character string is too long

A string of special characters is way too long (over 30 characters).

Macro Error Code 5

If without Endif was seen

Currently Vista macro processing must have an **EndIf** associated with each **If** statement, such as in the following example:

```
If a = b
  MessageBox("A is equal to B")
EndIf
```

If you attempt to code the executable statement on the same line as the **If** statement, you will get an error. For example, this is invalid:

```
If a = b MessageBox("A is equal to B")
```

See If Statement for more examples.

Macro Error Code 6

Unknown keyword seen

A keyword in the statement is invalid or unknown.

Macro Error Code 7

Unknown parameter seen

A parameter for the specified function is unknown. For example, the parm "txt" below was mistyped, and should really be specified as "text"

```
ReceiveFile("hostfile.text","pcfile.txt","txt")
```

For more information, see ReceiveFile or SendFile

Macro Error Code 8

Unexpected text seen

Some unexpected text was seen, usually following a valid macro line

Macro Error Code 9

While without EndWhile was seen

Currently Vista macro processing must have an **EndWhile** associated with each **While** statement, such as in the following example:

```
a = 1
While (a < 5)
  Debug(a)
  a = a + 1
EndWhile
```

If you attempt to code the executable statement on the same line as the **While** statement, you will get an error. For example, this is invalid:

```
While (a < 5) a = a + 1
```

See While Statement for more examples.

Macro Error Code 10

Variable or function not found

In Vista macros, variables do not need to be defined, but they need to be assigned to some value before they can be referenced. In the example below, variable "b" has not yet been set to any value, so error 10 is produced:

```
a = b
```

Macro Error Code 11

Unknown operator

An unknown operator was seen, such as the "<>" in the example below. See Operators for a list of valid operations.

```
If a <> 2
  Debug("a is not equal to 2")
EndIf
```

Instead, you might use the following for 'not-equal' (from the C programming language).

```
If a != 2
  Debug("a is not equal to 2")
EndIf
```

Macro Error Code 12

Unbalanced parenthesis

Some additional, or not enough parenthesis were seen, such as the following example:

```
a = ((2 + 3) * 4) + 5) * 6
```

Macro Error Code 13

Incompatible variable type for function

An attempt was made to evaluate string and numeric variables without conversion, such as the following example.

```
a = "12"
b = 10 + a
```

In this example, the string variable can be converted to numeric using the Val function before the addition is performed:

```
a = "12"
b = 10 + val(a)
```

Macro Error Code 14

Operation invalid for text variable types

An invalid operation was attempted on string variables, for example, subtracting one string from another. Only comparison and concatenation (+) operations are allowed.

Example of an error:

```
a = "string1" - "string2"
```

Examples of normal string concatenations and comparisions:

```
a = "string1" + "string2"

if a <= "string1string2"
   Debug("Looks ok to me"
Endif
```

Macro Error Code 15

Unknown function

An unknown function was seen, such as in the following example:

```
a = SomeFunction(12)
```

Macro Error Code 16

Invalid number of parms for function

Most macro functions require a specified number of parms. For example, the Str function requires a single number as input. The following is an example of an invalid function call:

```
a = str(12,13,14)
```

Macro Error Code 17

Invalid type of parm for function

Most functions require parms that are specifically string or numeric values. For example, the Str function requires single numeric parm. Calling with a string as in the following example, is invalid:

```
a = str("Hey there")
```

Macro Error Code 18

Operand seen where an operation was expected

An operation was expected between 2 operands

Macro Error Code 19

No more available file handles

Vista can open up 10 files at once. Any more than this causes an error. For example:

```
For i = 1 to 20
 handle = Open("c:\autoexec.bat","input")
 Debug(handle)
Next i
```

Macro Error Code 20

Invalid open-file action

Files can be open for INPUT, OUTPUT, or APPEND. Any other specification, such as in the following example, causes this error.

```
handle = Open("c:\autoexec.bat","read")
```

Macro Error Code 21

File open error

An error occurred when attempting to open the specified file. If input, most likely the file does not exist. You may need (for example) to specify the complete path, including disk letter, for the file name, such as:

```
handle = Open("c:\notfound.fil","input")
```

Macro Error Code 22

File is not open

The macro processing attempted to Get or Put to a file that is not open.

Macro Error Code 23

Record too long

The macro attempted to Put a string longer than 255 bytes to a file.

Macro Error Code 24

Invalid file handle

The specified file handle was not zero to 9.

Macro Error Code 25

End of file

The macro attempted to Get a record past the end of file.

Macro Error Code 26

Left parenthesis expected to follow function name

Functions in expressions, must have their parameters within parenthesis such as:

```
a = Str(123)
```

But other functions, such as Key and Type also require parenthesis, even though they are typically called as keyword statements rather than expressions, for example:

```
Type("PDF 2")
Key("Enter")
```

Macro Error Code 27

Invalid DebugWindow parm

The DebugWindow function is used to control the Macro processing debug window, and will accept only the Open, Close, Clear, and Hide strings as parms. See Debugwindow for more information.

If you are attempting to output debugging text to the debug window using this function, use the Debug function instead

Macro Error Code 28

Variable name is too long

Variable names can be up to 16 bytes long. The example below is an error:

```
a1234567890123456 = Str(12)
```

Macro Error Code 29

Reserved system variable cannot be altered

Read-only Variables cannot be altered in a macro. An attempt to do so results in an error, for example:

```
HostAddr = "100.12.45.01"
```

Macro Error Code 30

Invalid key function

An invalid key function was specified. See Functions for a list of valid key functions, such as the following:

```
Key("Enter")
Key("NextSession")
```

Macro Error Code 31

Timeout while waiting for condition to be true

The time specified on a Wait statement has expired before the specified condition occurred. For example, if you code the following:

```
Wait(20,Status="unlocked")
```

You are telling the macro processor to wait until the screen becomes unlocked, but wait for no longer than 20 seconds. If the screen remains locked for longer than 20 seconds, error code 31 occurs and the macro ends.

Macro Error Code 32

Variable table overflow

Vista has room to store about 4000 variables, either string or numeric. If you go over the limit, error code 32 occurs.

Macro Error Code 33

Cursor move key functions need row,col parms

Key functions such as MoveCursor or SelectWord, which depend on a mouse position, can be coded with row and col parameters if desired. See Key Functions for more information on this.

Macro Error Code 34

NextSession parm invalid

The Session function allows you to specify a session id letter, from "A" to "Z", indicating which session window to bring up. If the specified value is not A to Z, error 34 occurs. If the value is A to Z, but the specified session is not active, no error occurs and no action is taken.

To start a new session, issue the Key("NewSessionAsk") or Key("NewSession") function.

Macro Error Code 35

Else was seen without a previous If

See If for a valid example.

Macro Error Code 36

EndIf was seen without a previous If

See If for a valid example.

Macro Error Code 37

Internal error - invalid scan table address

Prior to starting a macro, the text is scanned to determine the location of If, Else, Endif, and other macro control statements. The results are placed in an internal "scan table". If this error message occurs, it is most likely caused by a Vista code error, not your own macro code. Please contact the author for help with the failing macro.

Macro Error Code 38

EndWhile was seen without a previous While

See While for a valid example.

Macro Error Code 39

Case was seen without a previous Select

See Select for a valid example.

Macro Error Code 40

Default was seen without a previous Select

See Select for a valid example.

Macro Error Code 41

EndSelect was seen without a previous Select

See Select for a valid example.

Macro Error Code 42

Next was seen without a previous For

See For for a valid example.

Macro Error Code 43

Assignment not seen on a For statement

See For for a valid example.

Macro Error Code 44

For statement was seen without a To parameter

See For for a valid example.

Macro Error Code 45

For statement must be all numeric values

See For for a valid example.

Macro Error Code 46

Step is the only statement that can follow To

See For for a valid example.

Macro Error Code 47

Value for Step cannot be zero

Since a For/Next statement uses the Step value to iterate the loop control variable, this value cannot be zero (otherwise an endless loop would occur). If you want an endless loop, use something like While(1) instead, and don't forget to provide a Break.

Macro Error Code 48

Next variable does not match previous For

See For for a valid example.

Macro Error Code 49

String variable too long (more than 255 bytes)

Sorry, string variables can only be 0 to 255 bytes long.

Macro Error Code 50

Break invalid outside a For/While block

See Break for a valid example.

Macro Error Code 51

Continue invalid outside a For/While block

See Continue for a valid example.

Macro Error Code 52

Scan area overflow. Program may be too large.

Prior to starting a macro, the text is scanned to determine the location of If, Else, Endif, and other macro control statements. The results are placed in an internal "scan table". If this error message occurs, it is most likely caused because your macro got a little bit too big.

Macro Error Code 53

Goto can only have a label

See Goto for a valid example.

Macro Error Code 54

Label not found

See Goto for a valid example.

Macro Error Code 55

File transfer parm error. See Help info for an example.

See SendFile or ReceiveFile for a list of valid parms.

Macro Error Code 56

Variable name is a reserved word or function name

Certain variable names are reserved for use only by the macro processor. See System Variables for a list of reserved variable names.

Macro Error Code 57

An error occurred while writing to a file

Some kind of error occurred while writing to a file. Sorry, Vista currently doesn't return the error code to the macro processor. Check for other system problems, such as a full disk or an I/O error on a device.

Macro Error Code 58

Get function cannot be used for file open for output

You attempted to read a file open for OUTPUT or APPEND with the Get function.

Macro Error Code 59

Put function cannot be used for file open for input

You attempted to write to a file open for INPUT with the Put function.

Macro Error Code 60

Clipboard is full

While writing to the clipboard via macro Put functions, the clibpboard became full (more than 65K bytes).

Macro Error Code 61

Invalid Parameter Value

The value specified on the SetOption function was not within the valid numeric range for that particular item. For example, the following will produce this error, because the maximum number of copy buffers is 9:

```
SetOption("Cut/Paste","NumberOfCopyBuffers",50)
```

Macro Error Code 62

Call function does not specify a subroutine address

You are probably attempting to Call a subroutine, but you forgot to indicate that the target label is actually a subroutine. The following code will produce this error:

```
Call check
Exit

check:   ;* forgot to code Subroutine here
  ...
```

Macro Error Code 63

Goto statement does not specify a label address

You probably coded a Goto statement, but the label on the Goto points to a subroutine by mistake. You can't branch to a subroutine.

Macro Error Code 64

Return seen outside of a subroutine

While scanning the macro, Vista saw a Return statement outside of a subroutine. It's possible you just forgot to code the Subroutine statement on the target label.

Macro Error Code 65

Attempt to call a subroutine recursively

During execution, Vista macro processing attempted to call a particular subroutine from within that same subroutine. Vista macros do not support recursive entry.

Support

Tech Support

Registered or Unregistered users can contact the author at **tom@tombrennansoftware.com** for problems, questions, enhancement requests, or anything else.

For problems, please give as much information about the problem you can, and also any of your own diagnostic results.

Troubleshooting

If you are having a problem, you may want to scan over some of these problems and solutions to see if you can resolve it yourself. Otherwise please read the Tech Support page. Please note that some of the proposed solutions listed below may seem obvious to you, but I guess I still have to mention them.

I can't get connected. I've never gotten connected yet using Vista.

Do you have a valid TCP connection setup on your PC? If not, please talk to your network folks and get that setup first.

Did you type in your host name or numeric address correctly?

If connecting using a phone line, is your connection dialed up and working properly?

Can you logon to the same host name using another PC, perhaps using another terminal? If so then possibly the problem is with Vista itself. Please send me a trace file if you don't mind.

If you are not sure of your target TN3270 server, try to logon to ibmlink.advantis.com. Even if you don't have a userid there, you should still be able to connect and get to their logon screen. If you can connect to ibmlink but not to your own server, then please send me a trace file of your attempts.

I'm connected ok, but my screen locks up when I try to logon, or when I do certain functions.

This obviously should not happen. There is nothing you can do but to send me a trace file, and I'll do my best to determine the problem and get a fix for you.

Why doesn't my window size change to what I set it to?

While Vista supplies a large set of raster font sizes, there still might not be one that fits your requested window size exactly, so Vista resizes the window to the nearest raster font. Sometimes this makes it look like the window resizing is broken. Try going to the **Font/Select Font** menu item and setting **Size Window to Font** off. This will leave the window size setting as you put it, but Vista may have to have some blank filler space around the screen area to make up for the lack of an exact font match.

Where is the Attention key, the Clear key, and others?

The default keyboard map in Vista is setup quite a bit like the IBM emulator. See the Default Keyboard Mapping help doc for help in finding a key, or better yet, Edit the keyboard to exactly the way you want it.

Even though I just installed Vista, it keeps telling me my trial period has expired.

Have you installed Vista on this particular PC more than a month ago? Perhaps it was installed once and then removed? Contact me for an extension code and you should be on your way to testing again.

If Vista has not ever been installed on this particular PC and you are getting the trial-is-over message, perhaps the date is not set correctly on the PC.

The toolbar icon with the picture of the clipboard doesn't do anything

The clipboard icon executes macro clipbrd.mac, and if you edit that macro you'll see it just does a WinExec function to call clipbrd.exe. Unfortunately clipbrd.exe (a nice clipboard viewer) is not always available, especially when running windows 95 or NT. If you can find a copy of this program on another computer, you can copy it to your windows directory. Otherwise just point the clipbrd.mac macro to another clipboard viewer. Last resort - use the toolbar editor to remove that button from the toolbar.

I've had enough of this so-called emulator, I want to remove Vista from my computer.

The easiest way is to hit Start/Control-Panel/Add-Remove-Programs, find Vista tn3270 in the program list, and double click to remove it from your system. If you have created any extra files in the c:\vista32 (default) directory, you'll need to delete those yourself. Also, you'll need to delete the vista.ini file in your windows directory for a complete cleanup.

Vista does not install any DLL's or anything else in your windows directories (other than the previously mentioned vista.ini file).

If you do remove Vista from your computer, the author would really appreciate a note about what you didn't like, or why it wouldn't work in your environment - if you get the time. Thanks.

Problem Documentation

If you came here, you must be really having trouble either getting connected, or with certain screen data being sent from or to the host. Please follow these instructions to aid in producing problem doc.

If you can reproduce the problem

Get to the point in your logon or processing a little before the problem is about to occur. Go to the **Help/Debug Functions** menu option, and select **Trace** so that it becomes checked. Vista is now tracing all input and output to a disk file named SESSIONx.TRC, where the 'x' is the current session window id (A, B, C, etc.). Check the title bar or **Window** menu option to see what session letter you are currently running.

Continue with your logon or processing until the problem occurs.

Close the Vista session window. If it is not possible to close the window, press ctrl-alt-delete and either kill the Vista task (win95 or NT) or reboot (win 3.1).

Before starting Vista again, rename or send the SESSIONx.TRC file to the author at tom@tombrennansoftware.com, with a note describing what you saw, what was wrong, what the screen looked like, or any other information you think might be helpful.

During the failure processing, with the trace file open, you may also periodically hit the **Help/Debug Functions** option **Dump Buffers**. This function dumps the current screen buffers to the trace file so the author can get an idea of before and after screen pictures.

Screen prints to a file also might be helpful to resolve the problem.

If you cannot reproduce the problem

Try to remember as much as possible about what you were doing and what the screen looked like when the problem occurred, and write that down in a note to send the author.

If you expect the problem to occur again, although intermittently, you might still turn on the trace function as described above. The trace file overhead is normally not noticeable, and the trace file will wrap around when it hits about 64K. So you can leave the trace running constantly, waiting for the error to occur again.

If the error does occur, stop Vista immediately and gather problem documentation as described above.

Registration

Vista is distributed as "Shareware", which is a method that gives you a chance to try out features of the program to decide if it meets your needs and works properly in your environment.

The Vista trial is distributed as a full, completely operational version, with a time limit of 30 days. About 20 days after installation, Vista will begin showing a reminder window with 2 registration options:

Register — This option brings up a registration form for editing. To register, just fill in the fields and mail it to the indicated address with your payment. Other registration details are on the form itself. Once registered, you will receive your personal Registration Code which will remove the time limits from the product.

Enter Code — Select this option when you have received your Registration Code. You'll need to enter the User Name and Reg Code exactly as listed on the registration note you received (if received by e-mail, use cut & paste for each field). Also, make sure you save your registration name and code, in case you change computers or hard disks and need to re-install the product.

Both these registration options can also be accessed via the Help menu item.

To keep the product as inexpensive as possible, nothing is mailed to the registered user except the registration code document. You already have the entire product installed from the trial distribution. Complete product documentation is supplied in the Help system.

For Registered Users

Thank you for the order! Your supplied User Name should now appear on the **Help/About** window.

If you received your registration code via e-mail, please send an acknowledgement back to the author at **tom@tombrennansoftware.com**.

Please retain your **User Name** and **Registration Code** in case you need to re-install Vista on a new computer or hard drive.

Thanks again, and I hope the product continues to be useful to you in your work.

Tom Brennan

tom@tombrennansoftware.com
www.tombrennansoftware.com

Restrictions

Vista was designed for a specific environment, and may not meet your needs in all aspects. Some of the items Vista does not support at this time are listed below, although support may be planned for the future:

General Items

Only standard 3278 models 2, 3, 4, and 5 are supported. Vista does not emulate 3279 graphics, or 5250. Just barely enough standard VTxxx terminal support is provided to get you logged on through a typical firewall.

Only TN3270 and TN3270E communication is supported. Vista cannot talk to any other type of host attachment device.

Any kind of HLLAPI interface is not supported

Setting up your PC as a VTAM LU for printing is not supported

Display Items

Extended field attributes of Field Outlining, Field Validation, Character Set, and Transparency are not supported.

Light Pen action is not supported

Notes from the Author

Hello,

I hope you find this version of Vista useful in your work. It was designed specifically for the mainframe environment, by me, an MVS system programmer who uses it every day. Here's some notes about emulators and related subjects in case you are bored enough to read them. You'll notice I talk in terms of the MVS mainframe and it's associated products, since that's what I do for a living. If that's what you do too, then you might find some helpful ideas within this text.

Tom Brennan

A Bit of History

Since 1983 I've worked with MVS mainframe computers as a System Programmer, and from that time until 1991 I used, almost exclusively, a real 3278 mod-4 dumb terminal. Of course, long before 1991 PC's had already been replacing real terminals in our MVS shop. But my mod-4 was about the last to go. For years I had seen my colleagues struggle with the emulators of that era. Typically they had large, difficult-to-read text, nobody used NewLine, Attention, or EraseEOF because they couldn't find the keys, and worst of all, the emulators slowed you down. I remember even in 1990 watching emulators slowly paint their text and redraw icons, while my trusty mod-4 was still plugging along doing instant screen updates.

Then came the 486 with an 800x600 NEC monitor. The first time I saw that setup I was hooked. The machine ran *fast*, the clear graphics painted almost immediately, and the options and functions available on the new OS/2 Communication Manager, such as type-ahead and multiple session windows, were too good to pass up. I finally said good-bye to my old mod-4, got a new 486-33 on my desk, and ran happily with that for many years.

Then in 1997 word came down from the top that we were to abandon OS/2 and standardize our desktops with Windows 95 - a logical move, actually. More than 80% of the company was already on some version of Windows and running a commercial tn3270 emulator. I personally have no real preference toward either OS/2 or Windows, but the mandate to convert to Windows meant I had to abandon Communication Manager and use another emulator that I did not like at all.

What better place to look for an emulator in 1997 than on the World Wide Web? That's what I did, and I found a commercially available one I liked it a lot. Only $150 and I was up and running with a first-class program, a knock-off of the McGill University emulator. Within months after buying it though, I saw some minor problems, and I had some ideas for improvements that I would like to have seen implemented. I wrote notes to the company, but they must have been a bit too busy to respond to a lone user. A new version came out with one new function I had requested, but in the process of programming it they had introduced a related bug. The frustration of that, plus the fact that as a continuous emulator user I was in a position to think of new useful features I could really use, I decided to write my own.

Vista is the result, and I hope it works for you as well as it does for me.

Usage Hints

Since I am the person who uses Vista the most, I thought I might try to pass along some general ideas I've learned while using it, and see if they might fit into your own Windows emulator environment.

Get a Standard 101 Key Keyboard

After struggling for a few weeks with one of the latest Windows 95 keyboards (the one with the new special keys), I went back an old 1984 IBM PC keyboard. The main reason for this at the time was because, as most mainframers do, I was using the right CTRL key as my Enter key. On the new Win95 keyboards, the right CTRL key is so small my finger was missing it's target. But upon going back to the 1984 keyboard, I also decided on one major change, which is my next recommendation below:

Make the ENTER key your ENTER key

If you only use your computer for 3270 emulation, then be that way. You can keep the right ctrl key as your Enter key and the Enter key as a NewLine key. But if you are like most people, you use other PC programs like MS-Word, Lotus Notes, etc., and you are constantly having to switch your brain as to the location of the Enter key depending on whether you are running the emulator or another PC program.

Rather than switch your brain, switch your keys. A NewLine key which is available without having to hit ctrl, alt, or shift is still a *must*. For example, when deleting an entire screen of members on an ISPF member list you would type d<newline>d<newline>d... for as many members as you have. If you had ctrl-enter set to NewLine (as many emulators default to) then you have to press d<ctrl><newline><lift ctrl> for each line. That's too many keypresses for me, and too easy to hit the <ctrl>d key in the process, which might perform some other function by accident.

The best way I've found is to get a keyboard with a large backslash key above the Enter key. In MVS work, the backslash is rarely used, so it can be edited to the NewLine function. Edit the <ctrl>\ key to a real backslash so you can still type the character the 3 times you will need it in your MVS lifetime. Then, most importantly, make the Enter key issue the Enter function, and get used to using it. You can still leave the right ctrl key as Enter if you want, but try not to use it.

Within a week of doing this, you'll be used to it and you'll notice a more seamless transition from your emulator into other PC applications as a result.

Use a Mod4 Terminal

Many people in our MVS shop still don't know the benefit of having 43 lines on the screen at once. With today's 1024x768 and larger displays, you should easily be able to find a Vista font that will be readable using Mod4 terminal emulation. You can get more done in less time. Your boss will be happy.

Just don't forget to check any ISPF panels you create with a mod2 session, so that you don't accidentally create panels more than 24 lines long.

Learn to Cut and Paste Without Thinking

Well... at least with minimal thinking. Vista has a rich set of cut and paste options, including items to select a single word, a field, or a line with a single keystroke or mouse click. The most useful of these might be SelectJCL, which is by default set to the right-mouse-click. It's a "smart" selection method that can quickly select strings and portions of strings based on mouse position and screen content.

Also, other various new ideas are supported with Vista such as the PasteRepeat function which will copy the clipboard data to the current field and also repeat it down the screen for as many

lines as there are on the screen - essentially doing things like the multiple <newline>d<newline>d... described above.

Vista also supports multiple cut/copy buffers, and overlays full screen input fields properly, so you can really make use of the Cut and Paste facility and save lots of typing and possible typing errors.

Know Your Keyboard, and Use It Effectively

I see many emulator users hitting tab 5 times to get to the next line, when a single NewLine keypress would have sufficed. I see people holding down the space-bar and letting it repeat until the end of a field because they don't use (or can't find) their EraseEOF key.

Position these keys and other essential ones like them in places where they are easy to get to. Don't worry about replacing other PC keys that you don't use with your mainframe work. For example, my personal keyboard setup uses the key to the left of the number 1 on the keyboard as my EraseEOF key. I rarely (if ever) need that character on the mainframe, and if I do, I the <ctrl> version of the key can still type it if necessary. But when set to an EraseEOF key, it's used *very* often. Take a look at your keyboard. The keys that are nice and clean on top are the ones that are used often. Find the dirty ones and change them to something useful so they'll get clean too. Just remember to wash your hands afterwards.

Also, you might think about running over to the Windows Control panel and setting the Keyboard Repeat Delay and Repeat Rate to their fastest values, or at least faster than the Microsoft defaults. You'll find this makes you feel the PC is just a bit faster, which then makes you work harder, which then gets you that raise and sends the kids to private school.

Keep the ctrl and alt functions for alphabetic and numeric keys to a minimum. That way you don't accidentally perform unintended functions in the middle of your edits when you hold a shifter key a little longer than you intended.

Use the Mouse MoveCursorEnter Function in ISPF V4.x

In ISPF V4.x, many panels are setup with a simulated graphical menu-bar at the top of the screen, and with other 'point and shoot' fields at strategic locations. With the left mouse button set to move the cursor and press Enter on doubleclick, you can then doubleclick these fields and quickly go places at speeds you've never gone before (well, at least a little bit faster anyway).

Use Macros for Repeated Edits

In the ISPF editor, you have the luxury of being able to issue commands such as, for example, C SYS1. BACKUP. to change a bunch of dataset names from one thing to another. But sometimes you need to do things like this repeatedly on other full-screen dialogs that have no change commands. If you find yourself doing this, push the Record button and edit one item, push Stop to save the macro, and then issue the macro as many times as needed to edit all entries needed.

The best way to do this is to have 2 or 3 macro buttons on the toolbar pointing to something like user1.mac, user2.mac, etc., and reuse these temporary macro files as needed. Even easier than hitting the toolbar is to edit a few of the numeric keys to issue the same temporary macros, and use those keys.

Also, once a macro is assigned to a toolbar button, you can right-click the mouse on top of that button to easily record, edit, etc. without having to hunt through a list of macro names.

Put the Julian Date and 24 Hour Time on the Status Bar

Since these 2 formats are often used on the mainframe, it can save you from having to find the Julian date on a calendar. And if you're like me, I can never figure out if 4pm is 14:00 or 16:00 since I seem to have missed the fun of military life, and can't add or subtract without a

calculator.

For example, the following variables in the Status Text line:

```
%m/%d/%y.%j %H:%M
```

... will resolve to something like:

```
12/19/97.353 21:56
```

Of course, you can put anything else you want on the Status Text line or on the Window Text line, as long as it will fit. See Variables for a list of other available variables you can use on the status line or window title bar.

TN3270E notes

The original TN3270 protocol, sometimes called "traditional tn3270" and described in RFC1576, had some items not addressed, such as:

- Attention and Sysreq functions had no specification
- WSF (write structured field) capability was not clear at connection time
- 3270 class printers could not be emulated
- There was no way to pass an LUNAME to the host
- The host BIND information could not be passed to the emulator

That first item (Attention and Sysreq functions) was probably the most important item to mainframe folks. With traditional tn3270, servers supported the functions in various ways, not always compatible with all terminal emulators. But for the past few years at least, it seems almost all tn3270 servers are consistent with regard to Attention and Sysreq handling, so this is not really a problem anymore.

Also in traditional tn3270, the second item (WSF ability) is now handled by a specification at connection time, and although not actually in the RFC standards, like the Attention and Sysreq handling it has essentially become part of the standard tn3270 server.

So... why am I saying all this? You might have noticed that Vista defaults to traditional tn3270 rather than the newer TN3270E protocol. This is because I've found the traditional method to work well in most situations, and avoids the extra header and acknowledgment commumincation required by the TN3270E protocol (see RFC 1647).

Also, Vista currently does not support 3270 printer emulation, and doesn't care about the BIND information passed from the host, so there really are no benefits of using TN3270E protocol unless you need to connect to a specific LUNAME, or you are having trouble with the traditional tn3270 connection.

Acknowledgments

The author wishes to gratefully acknowledge the following MVS System Programmers who helped make Vista what it is today:

Skip Robinson — User number 1. Many thanks to Skip for putting up with all the problems of a new program, and especially for all the suggestions and creative ideas, many of which are in the current product. Also thanks for having confidence in the product and sticking with it regardless of any problems encountered.

Tom Dien — User number 2. Originally asked to verify Vista on NT, but he ended up testing it down to the smallest details - reporting anomalies, making suggestions, and demanding perfection. Thanks for all the late-night e-mails, and the continuous re-installs required during beta testing. And thanks for introducing me to cafe sua da.

Also many thanks to the dozens of people around the world who found and helped diagnose bugs, made suggestions, and helped with testing.

You know who you are... 8-)

Tom