Table of Contents: INET UTS60

Installing the Inet UTS60 Emulator for Windows

System Requirements

The **Inet UTS60** emulator requires:

A 386 class PC or better with 4 MB of RAM and a hard disk.
Windows 3.1x, Windows 95 or Windows NT.
A Winsock 1.1 compliant TCP/IP stack such as provided by **Inet for Windows**, a third party or as part of the Windows operating system.
The TCP/IP service must be set up, configured and tested with ping.
A working IP network connection to a Unisys (Sperry) host.

Installation Procedure

Run the SETUP.EXE provided. (OK, if you didn't know this, you wouldn't have been able to read this help file .... ! Here we aim to provide you with instructions that would help future network wide installations.)

Run the UTS60 icon and check that all your settings are as required by your mainframe.

SETUP will add a command in AUTOEXEC.BAT to set the WINET environment variable to point to the install directory. In order for the command to take effect, the system must be rebooted. In the case of Windows NT the variable must be set manually using Control Panel. The WINET variable is used to find the licence file, DONTWIPE.ME!, and initial setup files such as the .INI files and the hosts file, INET.HST (which may be changed to any valid hosts file from the **Setup** or **Open** menus). To enable you to use the emulator before setting the WINET variable, the DONTWIPE.ME! file is also copied to the \INET directory in the root of the current drive. This copy can be removed once the WINET variable is set.

Uninstall Procedure

To remove UTS60 from your system, do the following:

1. Delete the WINET program group using the **Del** key or File Manager option **File/Delete** when the closed program group is selected.
2. Delete the WINET directory with all the files in it and also the \INET directory.
3. Delete INETUTS.INI and INET.INI in the Windows directory.
4. Edit the AUTOEXEC.BAT to delete the SET WINET statement.

Optional:

5. Uninstall the INET font (no harm if you keep it) from the **Control Panel** option **Fonts**.

Setting Up Many Users

In large file server based installations maintenance is much easier with all the users running UTS60 from a file server. The users will only require one or a few UTS60 icons on their own machines which can be set up quite easily. Another very useful part is the font that needs to be installed for every user. The supplied font, INETFONT.FON, gives a character size to fill the screen width exactly using a display mode of 640x480 as well as other sizes for 132 columns and 800x600 screen resolution.

The install script is an ASCII file, INSTALL.MST, which may be modified to suit specific requirements.   The number of icons with their descriptions and command line options may be changed.   It is also easy to change the default settings in the .INI files.

A fast way to install only the default icons: Copy the install files to a network drive (e.g. x:\DISK1) and run SETUP.EXE from there, selecting the "Icons only" option.

To create icon(s) manually in Windows 3.x, use the Windows Program Manager's **File/New** option while in the required program group and fill in the relevant fields.   To install the INETFONT go to the Windows Control Panel, select the Fonts icon and click the "Add" button. This procedure copies the INETFONT file from an accessible disk to the user's disk.   Another fast way to install the font is to copy INETFONT.FON to the user's \WINDOWS\SYSTEM directory and add the following line to the [`fonts`] section of his/her WIN.INI:

```
            Inet Fixed Pitch=INETFONT.FON
```

UTS60 Setup

Several settings should be set for correct operation of the UTS60 terminal emulator. Some of the options are asked at install time. Due to the variety of host systems, more setup may be required. The most important information is set up from the Settings sub-menu.

To be able to set up your emulator you need to know the type of system, environment and application, the TCP port, your PID, etc. Some defaults are set: For example a DCP, the TCP connection is made to port 256 and for an HLC to port 102.

Note: The Save or Save as new type options in the Setup menu must be explicitly activated to save settings for future invocations of the emulator. The last host and port connected to, will also be saved on Save or Save as new type

The following menu items are available under Setup:

Colour	
Palette	
Display hex/normal	
Font	
Function Keys	
Host File	
Host Names	
Settings	Various important settings
Printing	Screen or Host printing
Save	
Save as new type	

If you change some of your settings and save it to your .INI file but the setting seems never to be loaded from the .INI file, you might be trying to change a setting that has been locked by the system manager. You can check in the **Help/About** box whether an override file has been used.
See also Configuration files for more information.

Help Procedures

Help is obtained by clicking on the **Help** menu or pressing **Alt-H** while in the terminal emulator window.

Context sensitive help can be obtained while in the menu system.   When in a terminal emulator the **F1** key is handled as a terminal function key and not a Windows HELP key. When a menu item is selected, **F1** once more becomes a HELP key.   Use the cursor keys to move to a menu item and press **F1** to get specific help on that menu or option.   Dialog boxes have a HELP button which also gives specific help on that dialog box.

This Help file was designed to be also a manual, allowing you to use the **,<** and **.>** keys to page through it.   Note that many pages are longer than your screen and will require **PageDown** to read all of it.   Some of the pages are generic to various **Inet** products and some may contain information not applicable to the product you are using now.

Configuration Files in Inet for Windows

All the **Inet for Windows** products can use one, two or three configuration files.

The terminal emulators can use up to four files:

1) If you had an old version or a DOS version of the emulator, the ***.CNF** file is read first from the WINET directory (or the directory where the environment variable WINET points to).
Inet will offer to delete the .CNF file when the information has been transferred to a .INI file.
2) **INET.INI** is read from the WINDOWS directory to get the file name of the emulator's .INI file. (The default file name for the UTS60 is INETUTS.INI.) This .INI file will then be read. This is also the .INI file where any changes will be saved.
3) For any settings not found in 2) above the **INET.INI** is read from the WINET directory (or the directory where the environment variable WINET points to), for the file name of the emulator's .INI file. This is to supply a jump start when the specific file or entries are missing in the WINDOWS directory.
4) Lastly the **INET.INI** is read from the WINET directory (or the directory where the environment variable WINET points to), for an override file that might have been defined by your network manager. This override file may lock selected settings preventing users from changing it.

See the **Help/About** box for a list of .INI files used.

Note: Changing any of the settings will **only** have effect until the emulator is exited. To make changes permanent, that is to write it to the .INI file, choose the Save or Save as new type option in the Setup menu. All setup information is saved to the INET.INI file and *.INI file of the emulator in the WINDOWS directory.

See also: Setup

Keyboard

The Setup/Function Keys option gives a current list of keys and their definitions and allows users to change it.

The keyboard for the Inet terminal emulators is handled somewhat differently than other Windows applications. Because a terminal is emulated, some keys do not perform the normal Windows functions, but rather act as terminal function keys. ('Function keys' in this sense include cursor keys and **Alt-**, **Shift-** and **Ctrl-** key combinations.)

This is specifically true of the **F1**, **F10** and **Alt-F4** keys.   The **Alt-Fx** function key combinations can be redefined.   Thus **Alt-F4** does not do the normal Windows exit.   Rather use **Alt-X** for the **Exit** menu item.   When in a terminal emulator the **F1** key is also handled as a "normal" terminal function key and not a Windows HELP key. When a menu item is selected, **F1** once more becomes a HELP key. In the terminal emulators some key combinations retain their Windows functionality. These are: **Ctrl+Esc**, **Shift-Esc**, **Alt+Tab**, **Alt+ Esc** and **Alt+Spacebar**.

The keyboard layouts are stored in .INI files in the Windows directory with the emulator name appended to 'INET' as the file name.   You can change the keyboard layout or mapping by reprogramming keys at the Setup menu item: Function Keys.   You must save the changes with **Setup/Save** or **Setup/Save as new** to make them permanent.

Due to the various requirements by different applications, one keyboard layout will not satisfy all users.   To change keys' definitions, you have to know what is required by your application.   Keyboard mappings for UTS60 gives a list of keys and their definitions.

The **Esc** key is also redefinable. (To redefine **Esc** use **Ctrl-[** in the 'Key' field).   The **Tab** key cannot be reprogrammed.   The two **Ctrl** keys are redefinable in the 3270 emulator.

Due to the hot keys for the menu items under Windows, you might have to select and set your own key definitions on different keys as those used in the DOS emulators. For example, if you use **Alt-E** to execute some of your own functions, it will always open the **Edit** menu before doing your function.   To only get the **Edit** menu, press and release **Alt** and then press **E**.   To only get your own function and not a menu function, you have to select a different key combination, or run the emulator with no menu bar (the **/n** command line option).

See also: Keyboard mappings for UTS60
See also: Setup

UTS60 Keyboard

The Setup/Function Keys option gives a current list of keys and their definitions and allows users to change it.

The keyboard mapping of **Inet UTS60** is quite flexible. Keys like **Alt-U** can be defined if no menu item for that combination exists. If a menu item does exist, the defined key will be executed first, and then the menu item.

The keyboard for the **Inet** terminal emulators is handled somewhat differently than other Windows applications. Because a terminal is emulated, some keys do not perform the normal Windows functions, but rather act as terminal function keys. ('Function keys' in this sense include cursor keys and **Alt-**, **Shift-** and **Ctrl-** key combinations.)

This is specifically true of the **F1**, **F10** and **Alt-F4** keys. The **Alt-Fx** function key combinations can be redefined. Thus **Alt-F4** does not do the normal Windows exit. Rather use **Alt-X** for the **Exit** menu item. When in a terminal emulator the **F1** key is also handled as a "normal" terminal function key and not a Windows HELP key. When a menu item is selected, **F1** once more becomes a HELP key. In the terminal emulators some key combinations retain their Windows functionality. These are: **Ctrl+Esc**, **Shift-Esc**, **Alt+Tab**, **Alt+ Esc** and **Alt+Spacebar**.

The keyboard layouts are stored in .INI files in the Windows directory with the emulator name appended to 'INET' as the file name. You can change the keyboard layout or mapping by reprogramming keys at the Setup menu item: Function Keys. You must save the changes with **Setup/Save** or **Setup/Save as new** to make them permanent.

Due to the various requirements by different applications, one keyboard layout will not satisfy all users. To change keys' definitions, you have to know what is required by your application. We tried to list all common escape sequences in the **'Network Manager's** Guide'. The Setup/Function Keys option gives a current list of keys and their definitions.

The **Esc** key is also redefinable. (To redefine **Esc** use **Ctrl-[** in the 'Key' field). The **Tab** key cannot be reprogrammed. The two **Ctrl** keys are redefinable in the 3270 emulator.

Due to the hot keys for the menu items under Windows, you might have to select and set your own key definitions on different keys as those used in the DOS emulators. For example, if you use **Alt-E** to execute some of your own functions, it will always open the **Edit** menu before doing your function. To only get the **Edit** menu, press and release **Alt** and then press **E**. To only get your own function and not a menu function, you have to select a different key combination, or run the emulator with no menu bar (the **/n** command line option).

Default UTS60 Keyboard Mappings as in 1996

UTS key	PC key	Program sequence
Back Tab	Shift-Tab	{Esc} z
Cursor to Home	Home	{Esc} e
Debug FCC's	Alt-T	{Esc} ! d
Delete in Display	Ctrl-Del	{Esc} C
Delete in Line	Del	{Esc} c

Delete Line	PgUp	{Esc} k
Display Tabs	Ctrl-C	Ctrl-C
End of line	End	{Esc} ! E
Erase Display	Ctrl-Home	{Esc} M
Erase to End of Field	Alt-End	{Esc} K
Erase to End of Line	Ctrl-End	{Esc} b
Erase Unprot Data	Alt-Del	{Esc} a
F1 - F12	F1 - F12	{Esc} ! F 0 1 - {Esc} ! F 1 2
F11 - F22	Shift-F1 - F12	{Esc} ! F 1 1 - {Esc} ! F 2 2
F21 - F22	Ctrl-F1 - F2	{Esc} ! F 2 1 - {Esc} ! F 2 2
Insert in Display	Ctrl-Ins	{Esc} D
Insert in Line	Ins	{Esc} d
Insert Line	Alt-PgDn	{Esc} j
Line Duplication	Ctrl-PgDn	{Esc} y
Message Wait	Ctrl-Gr-/ or Ctrl-F12	{Esc} ! F 0 0
Position down	Ctrl-DownArrow	{Esc} ! i
Position left	Ctrl-LeftArrow	{Esc} ! g
Position right	Ctrl-RightArrow	{Esc} ! h
Position up	Ctrl-UpArrow	{Esc} ! f
Print screen	Ctrl-P	{Esc} ! P
Scan down	DownArrow	{Esc} i
Scan left	LeftArrow	{Esc} g
Scan right	RightArrow	{Esc} h
Scan up	UpArrow	{Esc} f
Script	Ctrl-S	{Esc} ! S *.scr {0D}
SOE	Ctrl-Num5	{1E}
Tab Stop Set	Ctrl-Tab	{Esc} {09}
Transmit	Gr-Enter or Ctrl-X	{Esc} ! T
Unlock keyboard	Alt-U	{Esc} ! U

Note: {Esc} = 0x1B in hex, or 27 decimal
{1E} = 0x1E in hex, or 30 decimal
{09} = 0x09 in hex, or 09 decimal
{0D} = 0x0D in hex, or 13 decimal

The keyboard layouts are stored in .INI files in the Windows directory with the emulator name appended to 'INET' as the file name. Changes are made from the **Setup/Function Keys** menu and have to be saved with **Setup/Save** or **Setup/Save as new** to make changes permanent.

See also: Setup

Command Line Options

The operation of **Inet UTS60** can be customised by command line options. You can create various icons or shortcuts for different tasks. Command line options are specified at the icon or shortcut properties. Use **Alt-Enter** on the icon to access the properties (or by selecting the **File-Properties** menu option from the Program Manager in Windows 3.x or the right mouse button in Windows 95). To duplicate an icon before changing, use **F8** or press **Ctrl** while dragging the icon with the mouse to the new position. It is recommended to change the icon description or shortcut name to reflect the additional functionality.

Example:

IWUTS60.EXE /h=dcp /o=2023 /p=P25 /x will establish a connection to a server with the host name 'dcp' on port 2023 with PID 'P25' and exit the emulator when the session is closed.

See also Setting Options for some of these options that may be Saved in your .INI files.

Syntax

IWUTS60.EXE [[/h=]*HostName* [/n]] [[/o=]*port*] [/p=*PID*] [/t=*emul_type*] [/x] [/s=*script_file*] [/i=*inet.ini_file*] [/c[=]*old_.CNF_file*]

Parameters

/h=*HostName*

A default remote server host name may be specified on the command line. Inet UTS60 attempts to open the session immediately on entry. Even if the session is subsequently closed, only another connection to the same server host can be opened from the open menu item. The server host is specified by entering its name or IP address as an argument. The **/h=** is optional when specifying the host name as the first argument.

Example:

IWUTS60.EXE /h=dcp will establish a connection to a server with the name 'dcp'.

/o=port

If the default port is not 256 for DCP or 102 for HLC, the port may be specified as well. With **/o=** the port can be specified. Remember that the port number is saved in the .INI file.

Example:

IWUTS60.EXE /h=dcp /o=257 will establish a connection to a server with the name 'dcp' on port 257.

/p=*PID*

If the parameter **/p**= is given, the PID can be specified on the command line. See also the Settings Dialog Box where it is more often specified and saved.

Example:

IWUTS60.EXE /h=dcp /o=257 /x /p=P03000 /t=C:\MYINI\MYUTS will establish a connection to the remote server 'dcp' on port 257 with PID P03000 and will use settings from C:\MYINI\MYUTS.INI and will exit the program when the session is closed. [to Syntax line]

/t=emul_type

The type of emulator or .INI file to use is specified here. If you want to define a new emulator type with different PID, keyboard mappings and/or settings, you can load a new type with the /t option. The name is usually linked to emul_type.ini containing the required settings. Names that would result in invalid file names may be used if it is defined in the user's INET.INI file in the [Emulators] section. See also the Setup/Save as new type menu option.

Example:

Suppose you want a UTS60 with different keyboard mappings: Save the setup with the Save as new type option in the Setup menu, using '**MyUTS**' as the new type. All the settings and keyboard mappings will be saved to **MYUTS.INI** in the **Windows** directory. Create a new icon with an appropriate description and specify the following command line arguments: **IWUTS60.EXE /t=MyUTS**. This icon will run UTS60 of type '**MyUTS**'. [to Syntax line]

/x (Exit)

If the parameter **/x** is given on the command line, the program will exit when the session is closed. The **/x** parameter will only work if a remote host was specified on the command line with the **/h=** option.

Example:

IWUTS60.EXE /h=dcp /x will establish a connection to a server with the name 'dcp' and will exit IWUTS60.EXE immediately when the session is closed. [to Syntax line]

/n (No menu)

If the parameter **/n** is given on the command line, the emulator program will have no drop down menus. The **/h=*hostid*** option is then required. The **/n** parameter will only work if a remote host was specified on the command line as well. The **/x** option is implied.

Example:

IWUTS60.EXE /h=dcp /n will establish a connection to a server with the name 'dcp' and will have no menu and will exit IWUTS60.EXE immediately when the session is closed. The session has to be exited since with no menu, you can't do anything while still in the program! [to Syntax line]

/s=*script_file*

If this option is specified, the given script file will be executed. This is the recommended method to run an automatic log on script or batch jobs. If the name contains an asterisk (*), for example **'*.scr'**, a dialogue box will prompt you to make a selection. See also the Script Language reference and the Scripts menu item.

Examples:

IWUTS60.EXE /h=dcp /s=logon.scr will start the emulator, open a connection to 'dcp' and start the script file.
IWUTS60.EXE /h=buro /s=*.scr will start the emulator and prompt you to select a script file. The script file will then be executed. [to Syntax line]

/i=*inet.ini_file*

A command line argument starting with **/i=** is interpreted as overriding the default terminal emulator's configuration file of **\WINDOWS\INET.INI**. The argument **/i=C:\MYDIR\MYINI.INI** specifies MYINI.INI in the \MYDIR directory of drive C: as the configuration file. If the setup is saved, the emulator type and it's .INI file name will be saved to the specified command line .INI-file, or if it is not specified, to **INET.INI**. in the Windows directory. The override facility specified in WINET\INET.INI will still be used.
[to Syntax line]

/c=*old_.CNF_file*

For backward compatibility this option is intended to be used only once to read an old existing .CNF file. Use the **Settings/Save** option to save it as a .INI file which should be read with the **/t**= option.   The **/c=** option will also read the .INI file with the same name when the .CNF file has been deleted, but a warning will be displayed.   The argument **/c=C:\MYDIR\MYFILE.CNF** specifies MYFILE.CNF in the \MYDIR directory of drive C: as the configuration file. The CNF format is outdated and should not be used. The preferred way is to use .INI files.

[to Syntax line]

Inet UTS60 Menu Items

Exit	Exiting the emulator
Open	Opening a connection
Close	Closing a connection
Edit	Using clipboard functions
UTS60 Setup Menu	Setting up the UTS60 emulator
Read	Reading and sending files
Write	Writing or logging data from the server to disk
Editor	General purpose text editor
PrintScreen	Printing the current emulator screen
Scripts	Using scripts on the terminal emulator
Help/About	Useful information about configuration

Exit

Use **Exit** to exit from the terminal emulator.

If the connection to the server host is still open, you will be prompted with an option to close it or to abandon the **Exit**.

See also: Open, Close

Open/Close

Use **Open** to open a connection to the remote server host.

A dialog box prompting for the remote host name and the port to use, will appear.

To select a host from the host file INET.HST, open the drop down list and select an entry. Alternatively a host name or Internet address in dotted decimal format (for example 192.6.1.4) can be typed. If an unknown host is specified, the dialog box will not accept an 'OK' until a valid host is supplied.

Normally the port does not have to be specified and the default value of 102 for HLC or 256 for DCP is acceptable. If it is necessary to use another port, enter the value in the Port field.

Select 'OK' to accept the selections and to try opening the connection.
Select 'Cancel' to cancel the operation.

The dialog box will NOT be available if a host name has been specified on the command line with the **/h=** (host) option. The user will NOT be prompted for a host name or port number. It will not be possible to make a connection to another host or port number with this setup.

See also: Command line options

If the Setup/Save option is invoked after opening a connection, the host name and port number will be saved to the configuration file. The next time the terminal emulator is started, these settings will be the default selections.

See also: Close

Close

This option closes an open connection to a remote server host.

If the **/x** (exit) option has been specified on the command line, it will also exit the emulator program.

See also: Command line options

See also: Open

Edit

The Edit menu item is used to access the clipboard. The clipboard provides several features for copying and pasting text.

How to select text for clipping

Copy block/lines	Copying data to the clipboard
Paste	Pasting data from the clipboard
Copy all	Copying everything on the screen to the clipboard
Clear	Clearing the clipboard
Block/Line Marking	Setting the marking mode

The built-in text editor may be useful to paste text from multiple origins temporary.

How to select text for the clipboard

Text is selected by positioning the I-beam cursor to the start of the text to be marked, pressing and holding the left mouse button down while dragging the cursor to the end of the text to be marked and releasing the button.

Since most keys are used for sending data to the remote host, there is no way of selecting the text with the keyboard.

The way text is selected depends on the currently selected marking mode. Text is visually marked by displaying it in reverse video. Text already displayed in reverse video will not be changed.

Text is unselected by clicking the left mouse button without moving the mouse. Your text cursor will also be moved to the clicked position.

See also: Edit

Copying data to the clipboard

Text will be copied to the Windows clipboard when you select this menu item.

Before text can be copied to the clipboard, it must first be selected. If you have not selected anything, the option on the menu will be greyed.

If the marking mode is set for block marking, the menu item displays '**Copy Block**'. If the marking mode is set for line marking, the menu item displays **'Copy Lines**'

Selected text that was copied to the clipboard, can then be pasted into another application, or even sent to a remote application by pasting it in your emulator itself.

See also: Edit

Pasting data from the clipboard

If there is text data in the clipboard, it can be pasted into the terminal emulator.   If the clipboard is empty, the menu option for pasting will be greyed, indicating that there is nothing to paste.

Pasting data into the emulator means that characters are copied from the clipboard as if they have been typed from the keyboard. Each line of data is followed by a CR that may be translated into a CR-LF combination depending on the currently selected outbound translation mode.

You have to keep the above mechanism in mind when undesired things happen:

- When pasting more than one line from the clipboard, the starting position of the second line will be determined by your application handling the new line or CR at the end of the previous line.
- Pasting more data to a field than the size of that field will also have to be handled by your application.

See Setup/Settings for more information.

See also: Edit or the Editor option where text may be pasted temporary.

Copying All to clipboard

Selecting this item causes everything on the emulator screen to be copied to the clipboard.

See also: Edit

Clearing the clipboard

This option clears the entire clipboard.   It will not remove the marked block.   To remove the reverse video marking made by the mouse click-and-drag, click with the mouse any where in the emulator screen without moving the mouse.   Your cursor will also be moved to that spot.

See also: Edit

Setting the Marking Mode

Text can be selected in block mode or line mode.

If the marking mode is set for block marking, the menu item displays '**Copy Block**'. If the marking mode is set for line marking, the menu item displays '**Copy Lines**'.

Block mode enables you to copy a rectangular piece of text from your screen. This is useful for column marking or for copying a table of data to the clipboard.   A typical example is to copy a column from a spreadsheet.

Line mode is useful for text applications. Text is selected as full length lines from the starting position to the ending position.

The menu item for selecting the required marking mode either displays '**Line Marking**' or '**Block Marking**'. If '**Line Marking**' is displayed, the system is currently in block mode. This means that, if you select this item, it will change to line marking mode. The opposite is true for '**Block Marking**'.

Block Mode

The blue text is a block mode example.
Text will be copied in a block
A small table: 1234567   This is the end of the small table line 1 will continue here to the end
A small table: 1456789   This is the end of the small table line 2
This is the end of table
In block mode, only the digits highlighted in blue will be copied to the clipboard.

Line Mode

The blue text is a line mode example.
Text will be copied as complete lines.
A small table: 1234567   This is the end of the small table line 1
A small table: 1456789   This is the end of the small table line 2
This is the end of table
In line mode, only the text highlighted in blue will be copied to the clipboard.

See also: Edit

Setup

Several settings can be set here for correct operation of the terminal emulator. The most important information is set up from the Settings sub-menu.

The following menu items are available under Setup :

Colour	
Palette	
Display hex/normal	
Font	
Function Keys	
Host File	
Host Names	
Settings	Various important settings
Printing	Screen or Host printing
Save	
Save as new type	

Note: Changing any of the settings will only have effect until the emulator is exited. To make changes permanent (that is to write it to the .INI file), choose the Save or Save as new type option in the Setup menu.

If you change some of your settings and save it to your .INI file but the setting seems never to be loaded from the .INI file, you might be trying to change a setting that has been locked by the system manager. You can check in the **Help/About** box whether an override file has been used.
See also Configuration files for more information.

Colour

Use **Setup/Colour** to select the foreground and background colours of normal text.

A colour dialog with 16 foreground and 16 background colours is displayed. A box with sample text is also displayed and is modified to indicate the effect of the current selection. Selecting the foreground colour to be the same as the background colour obviously results in non-readable text.

After selecting a different background colour, the screen may not immediately show the new background correctly.   When new text is being written to the screen, the new colours will show.   After saving the new setting, exiting from the emulator and running it again, the colours will be correct from the start.

The way that bright or highlighted text is displayed depends on the colours selected in this dialog. If a bright colour is selected as the normal foreground colour, a dim colour will be displayed when the bright attribute is set. The same is true for the background colour. The default setup is a black foreground on a bright white background.

Notes:
Specifying a blue foreground with a red background and normal brightness results in dim blue characters on a bright red background.   While specifying the bright attribute results in bright blue characters on a dim red background.

To make the screen behave exactly like an equivalent terminal, a dim white foreground (the grey on the right hand side) on a black background can be used, but this looks less elegant than a white Windows background.

The 32 colours that are displayed can be changed by setting up the Palette.

See also: Adjusting the Palette, Setup

Adjusting the Palette

Each of the 32 colours that can be selected in the **Setup/Colour** dialog, can be changed here.

To change the palette of a colour, first select the colour to change by clicking on the desired colour. Now use the Red, Green and Blue scroll bars to change the intensity of each colour component.

In the middle of the Palette Dialog is an example of how the colour will appear in the emulator session. Click on a background colour and then on a foreground colour. The colour that was last clicked, will be the one that can be modified.

You can click on 'Reset' to reset the colour to the Windows Driver default. Clicking on 'Reset all' will reset all colours to their defaults.

Notes:
Some VGA systems cannot show more than 16 colours under Windows 3.1. The Windows driver will then attempt to match to the nearest colour by dithering the colours. Although the colour is dithered, the example box will still show how it will appear on the screen.

When you change from 16 colour to 256 colour mode in Windows, or the other way round, it may be necessary to use 'Reset All' to reset all colours to the new driver defaults.

See also: Colour, Setup

Display hex / Display normal

Do not use **Setup/Display Hex** or **Setup/Display Normal** to switch between normal display with escape sequence interpretation and hex display with no escape sequence interpretation.   This option is only supplied to assist in debugging applications being developed on your host system.   It is a nice way of messing up your screen for normal users.

See also: Setup

Font

Use **Setup/Font** to choose another font to be used for characters displayed on the emulator screen and/or to re-size the screen to optimal size.

The font type as well as the font size can be specified. You are restricted to choosing a fixed-pitch font, because terminal emulation will not work correctly with a proportional font. The text cursor, for one thing, will not be positioned correctly.

After the font has been selected, the emulator attempts to adjust the window size to fit the entire emulator screen without using scroll bars. If the resulting window is too large to fit onto the screen, scroll bars will be used.

See also: Setup, Colour, Palette

Function Keys

Use **Setup/Function Keys** to view or change the character sequences generated by function keys. 'Function keys' in the **Inet** emulator include cursor keys and **Alt-**, **Shift-** and **Ctrl-** key combinations. Do not confuse the actual terminal's function keys with the Inet emulator's function keys. A terminal normally has special keys to perform special functions. You may map any of those special functions to an Inet function key. You may also define macros of up to 15 character long on any **Inet** function key.

To view the current keyboard layout:

When the dialog box appears, you can press any function key/combination and see the terminal key or function linked to it. The escape sequence activating that function is displayed in ANSI and in hexadecimal in the 'Text' and 'Hex' fields. There are more than 200 programable function keys, so most of them will be open.

Click on the terminal key drop down list to see the terminal keys or functions. The PC key field will show which function key is linked to that terminal key or function. Click on the **Find next PC key**' button to see if any other keys has been assigned to this terminal key.

To change an assignment:

1. Mark the '**Enable change**' tick box
2. Click in the PC key field
3. Select the terminal key from the drop down list.
4. Click on OK or go to the PC key field to select a new function key.
5. When you are happy with your changes, select the **Setup/Save** option from the menu bar to save your changes to an .INI file.

In stead of selecting a PC key or function, you may enter your own string or escape sequence in the 'Text' and 'Hex' fields. The 'Hex' field is used to enter non-display codes in hexadecimal for example the **Esc** code is 1B in hex. Non-display codes are displayed as a thick vertical bar character in the 'Text' field. When making changes in one field, the other fields will be updated to reflect the latest changes.

Use the 'Restore' button to undo all changes made during the life of this dialog box.

Warning: In '**Enable change**' mode, it is quite easy to redefine keys. If you scroll through the terminal keys or change the 'Text' or 'Hex' fields and then select a new PC key, the previous key will be assigned whatever was displayed. To return to the previous setup, just click on the **Restore** button, or exit the emulator without selecting the **Setup/Save** option.

Note: Entering binary zero or NULL codes must be done in a special way. Enter them in the 'Hex' field only and do not switch back to the 'Text' field before accepting the definition.

Note: Changing any of the settings will **only** have effect until the emulator is exited. To make changes permanent (that is to write it to the .INI file), choose the Save or Save as new type option in the Setup menu.

Various things to note:

Due to the various requirements by different applications, one keyboard layout will not satisfy all users. To change key definitions, you have to know what is required by your application.

The **Alt** key combinations cannot be used in some emulators with the alphabet keys. The **Alt-Fx** function key combinations can be defined. Thus **Alt-F4** does not do the normal

Windows exit, rather use **Alt-X** for the **Exit** menu item.

The **Esc** key is also redefinable.   The **Tab** key cannot be reprogrammed.   The two **Ctrl** keys are redefinable.

Due to the hot keys for the menu items under Windows, you may have to select and set your own key definitions on different keys as those used in the Inet for DOS emulators. For example, if you want to use **Alt-E** to execute some of your own functions, it will always open the **Edit** menu before doing your function.   To only get the **Edit** menu, press and release **Alt** and then press **E**.   To only get your own function and not a menu function, you have to select a different key combination, or run the emulator with no menu bar (the **/n** command line option).

See also: Keyboard mappings for UTS60
See also: Setup

Host File

The Host File is normally the file used by your TCP/IP implementation to link symbolic host names to IP addresses.   The **Inet** host file is normally INET.HST.   The Microsoft WinSock use a file called HOSTS that is often not even created.

The host file can be used to define the addresses of hosts often used.   The next option, Host Names, can be used to edit the host file.

See also: Setup

Host Names

The Host Names option starts an editor to edit the Host File.

See also: Setup

Settings Dialog Box

Use Settings to set various terminal settings.   Many of reasons for failed connections are incorrect entries in these fields.   Please check with your system manager that you have the correct ID's and settings here.

A dialog box divided into five sections is displayed:

PID	Define the PID
System	Define System Name
Environment	Define the Environment
HLC ID	Define the HLC ID
HLC	Select HLC or DCP mode
Uppercase	Select uppercase mode for all input
Rows	Set screen size: number of rows
Columns	Set screen size: number of columns
Read mode	Select file read mode as binary or text
Write mode	Select file write mode as binary or text for logging a session

Note: Changing any of the settings will **only** have effect until the emulator is exited. To make changes permanent (that is to write it to the .INI file), choose the Save or Save as new type option in the Setup menu.

If you change some of your settings and save it to your .INI file but the setting seems never to be loaded from the .INI file, you might be trying to change a setting that has been locked by the system manager.   You can check in the **Help/About** box whether an override file has been used.
See also Configuration files for more information.

PID Definition

The personal ID (PID) can be entered in the PID field. This will be the ID of the workstation which will be sent to the remote host.

Note: The PID can also be set with an environment variable.   For example:
If the environment variable **'NEWPID'** is defined with the DOS command **'SET NEWPID=P03000'** and you specify **%NEWPID%** as the PID, it will be set to **P03000**

System Definition

The name of the system to run on the remote host. The default is '**MAPPER**'.

Environment Definition

This is the environment on the Unisys in which the UTS60 will operate.   For example: **tip**, **demand**

Define HLC ID

Change this setting when connecting to a Host Lan Controller requiring a different ID.   The default ID is TIPCSU.

Select HLC/DCP Mode

Select this setting when connecting to a Host Lan Controller.   The default is a Data Communications Processor connection.

Uppercase Mode

With this setting enabled, all keyboard input to UTS60 will be in upper case, regardless of the Caps Lock or Shift state of the keyboard.

Rows

This field specifies the number of screen rows. The maximum value is 50 but the usefulness of larger screens will depend on your application.   If your terminals is running only in scroll mode, you may use larger screens, allowing you to scroll back to previous screens using the scroll bar.   A too large screen may cause corrupted display for applications that use relative or absolute screen addressing.

Use **Setup/Font** to re-size the screen to optimal size.

Columns

This field specifies the number of screen columns. The maximum value is 132 but the usefulness of larger screens will depend on your application.   Some emulators are switched automatically by escape sequences sent from the application when needed.

Use **Setup/Font** to re-size the screen to optimal size. .

Read mode

The Option Buttons (also called Radio Buttons) select the mode that a file to be read is opened in - either Binary or Text.   This mode will be used when input comes from a file opened with the **Read** menu option.

Write mode

The Option Buttons (also called Radio Buttons) select the mode that a file to be written is opened in - either Binary or Text.   This mode will be used when creating a log or trace file with the **Write** menu option.

Printing

You can set up two printers for two types of printing: Select '**Screen Printing**' for the printer where screen copies (dumps) are to be printed. Only if your mainframe sends print jobs imbedded in the data stream (print through) you may select '**Host Printing**' for the printer where jobs scheduled from the mainframe are to be printed. The two printer definitions may specify the same physical printer. (Jobs sent from the mainframe to an LPD or socket printer would require the right *print server* to be started as a separate program. See the **Inet LPD Server** and **Inet Print Server**.)

The '**Screen Printer Setup**' and the '**Host Printer Setup**' dialogues are identical. You first have to select an installed printer. If there is no suitable printer driver available, you will have to use Windows Print Manager to install the correct drivers for your printer.

Inet supplies a driver option, '**Print to file on disk**', which allows the print data to go unmodified to a file (or straight to the printer port). When you have selected '**Print to file on disk**' as your printer, the '**File Name**' field specifies the print destination. To print directly to a printer port, specify the file name in the '**File Name**' field as 'LPT1' or 'LPT2'.

The '**Setup**' button will call the setup dialog of the driver for the selected printer.

The '**Font**' button will allow you to select a printer font if supported by the printer driver. (**Hint:** Select fixed spacing fonts only for better spacing on printouts that contains tabulated data. The Courier font is the best example of a fixed spacing font.)

The '**Device font**' check box will disable font selection and use the default font of the printer device.

When '**File Name**' is specified, printing is redirected to the specified file or device. To be prompted for a file name when the print job starts, leave this field empty and select your printer as '**Print to file on disk**' or a printer driver connected to FILE in stead of LPT1. To print directly to a printer port, specify the file name as 'LPT1' or 'LPT2'.

When printing for the first time to a file in a session, the '**Append**' check box can be marked to append to the file of previous session. During a session, it will always keep on appending, unless you change the file name. This setting is only used by the '**Print to file on disk**' printer. Printer drivers which are printing to a file, will always overwrite the file.

The '**Convert LF to NL**' check box will, if checked, translate a single line feed character (LF) received from the remote server host into the two characters new line (CR-LF) sequence.

The '**Delay**' field is used to decide when a print job is completed. It should be set to a value (in seconds) which is longer than any interval during a print job when no data is received, but more is still forthcoming before the end of the print job. The default is 10 seconds which means any printout will start 10 seconds after the last data was received. If it is too short, you may get the contents of a single page spread out over more than one page, if it is too long, you may get two or more pages from consecutive jobs concatenated onto one page.

See also: PrintScreen which should be used to test your printing configuration.

Printer Font

In the '**Select Printer**' dialog, the '**Font**' button will allow you to select a font if supported by the printer driver.   The last characters on an 80 column display might disappear if the font is too large.   If your printer or printer driver does not support the selected font, unpredictable results might be produced.

Hint: Select fixed spacing fonts only for better spacing on printouts that contains tabulated data.   The Courier font is the best example of a fixed spacing font.

Save

This menu selection will save the current set-up parameters to the configuration (.INI) file to make them permanent.

The default configuration file is INETUTS.INI in the WINDOWS directory. Along with all the information which can be changed in the Setup menu, the last host name and port number used in Opening a connection is also saved.

If your system manager has enabled the Override feature, certain setting changes will not register.   You can check in the **Help/About** box whether an override file has been used.

See also: Setup and Save as new type

Save as new type (Define a New Emulator Configuration**)**

Function key maps, colours and many settings of the emulators are saved in .INI files.   You may change these settings and save them to define a new terminal emulator.

Load the emulator and modify the required setting under the Setup menu.
Save the new settings in a new file by using the **Save as new type** option.
Copy the emulator icon by pressing **Ctrl** while dragging it with the mouse to a new position.   (You can also use **F8** or the Program Manager **File/Copy** menu to select the group where you want the new icon.)
Press **Alt-Enter** to edit the icon properties (or the Program Manager **File/Properties** menu).
Modify the description to reflect your changes or purpose with the new emulator.
At the end of the command line you may add any of the options described in Appendix A.
E.g.:
Add a ***/t=emulator_type*** string to use the newly defined colours or keyboard layout.
If this emulator is to be used to a specific host you normally will specify the host to which this emulator must connect on the command line with the **/h=*host_id*** option.
The **/x** option may also make things more convenient by closing the emulator when the session to the remote host is closed.

See also: Setup

Read

Use READ to read a text file and send it to the remote server as if it had been typed in.
A file dialog will be displayed allowing you to select a file to send. Use OK or press Enter to start the operation.
The READ operation can be aborted by pressing any key while the file is being sent.

Write

Use WRITE to write all characters received from the remote server to a local file. A file dialog will be displayed allowing you to specify the name of an existing or new file. If an existing file is selected, you will be prompted to overwrite it or append to it. The mode the file will be opened in (binary or text) is determined by the Setup option: Settings.

Editor

The editor provided in the emulators is a small, general purpose editor.

It can be used for whatever editing purpose, but its main intention was to edit system or configuration files (like scripts and .INI files) without leaving the emulator.   It is also useful in multiple cut-and-past operations, to store or accumulate data.

Note: The editor supports cut-and-paste, as well as search and replace

See also: Clipboard functions

Print Screen

Use the **PrintScreen** menu item to make a copy of the text on the emulator screen only. The printout is sent to the printer as specified in **Setup/Printing**.

It is also possible to program a function key to do the print screen: Use the escape sequence as specified for your terminal type.   For an UTS60 it is **Esc ! P**.   (The **PrintScreen** keyboard key is still controlled by Windows and will do a graphics screen dump to the clipboard as specified in your Windows operating system.)

Running Scripts

You can use the **Script** menu option to launch a script at any time during an active session.
A dialog box will ask you for the script file and start to run it when you select the file.
The recommended way to start scripts that should do an auto-logon, is to specify the script on the command line of the icon or shortcut.
It is also possible to program a function key to run a script: Define a key as '**Esc ! S** scriptfile.scr CR'. (Use no spaces and CR is 0D in hex. See Function Keys)
See also the Script Language reference section.

Help/About

Use the **Help/About** option to display licence, current configuration, copyright and revision information.   Up to three .INI files may be listed that were used to obtain the settings loaded for the current session.

The first file listed, called "Initial" might indicate the INET.INI file in the WINET directory.   The "Initial" file is used to find information not available in the second, "Current", .INI file.   The "Current", .INI file is also where any changes will be saved.   The last file, "Override", might indicate a .INI file set up by the system administrator, that overrides user changes saved in the "Current" .INI file.   The override facility is described in the **Inet** "Network Manager's Guide".

Script Language

In **Inet for Windows**, scripts can be run by specifying them on the command line with the ***/s=script_file*** option (for automated login and batch tasks) or from the **Script** option on the menu bar.

The **Inet** script language is documented under the following headings:

List of Keywords :

To find the specific keyword descriptions see the paragraph given next to the keyword.

Topics:

- **operators** - see Numeric Operators or String Functions
- **pause** - see **wai**t
- **read** - see **screen, key** or **readln**
- **remark** - see Comments //
- **s0** - **s19** - String Variables, see Variable Types
- **write** - see **display, send** or **writeln**
- **v0** - **v19** - Numeric Variables, see Number Formats and also Variable Types

Keywords:

- **close** - File Functions
- **col** - Input and Output Functions
- **debug** - Debug Options
- **display** - Input and Output Functions
- **else** - Program Flow Control
- **endif** - Program Flow Control
- **endwhile** - Program Flow Control
- **execute** - Start Other Programs
- **ferr** - File Functions
- **find** - String Functions
- **flags** - Input and Output Functions
- **if** - Program Flow Control
- **instr** - String Functions
- **key** - Input and Output Functions
- **len** - String Functions
- **match** - String Functions
- **mid** - String Functions
- **open** - File Functions
- **readln** - File Functions
- **row** - Input and Output Functions
- **screen** - Input and Output Functions
- **send** - Input and Output Functions
- **str** - String Functions
- **val** - String Functions
- **wait** - Input and Output Functions
- **while** - Program Flow Control
- **writeln** - File Functions
- **xlate** - String Functions

The Script Interpreter

The script interpreter is active while a script file is executed. A script file is a text file with commands which are used to automate the operation of the terminal emulator. The script interpreter can manipulate keyboard input as well as data displayed on the screen, wait for an event e.g. keyboard input, host output or a time-out, simulate keyboard input, read or write a file and execute another program. A script file can be executed by specifying it on the command line with the ***/s=script_file*** option,   assigning it to a function key or by starting it from **Script** on the menu bar.

An editor is included in the emulator which can be used to edit script files.   You can access the editor from the **Editor** menu item.

When a script is executing, keyboard input is intercepted and passed to the script interpreter, which can process it in any way including passing it on unchanged. The script language provides program flow control statements **if ... else ... endif**, **while ... endwhile** and **wait.**   It supports 16 bit signed integer expressions and string expressions including functions to convert from one format to the other and the ability to use special values like the cursor position and screen contents.

Comments

Comments start with // anywhere on a line and are ignored. Comments and empty lines will not slow down execution of the script nor use memory and should be used liberally to document the script program.

Number Formats:

Decimal numbers are typed just as they would normally be written. The **range** for decimal numbers is from -**32768 to +32767**.

For example: 123 means "one hundred and twenty three".

Note: To assign a negative number to a variable, you **must** subtract the value from zero. For example: **v3 = 0** - **12** assigns the value -**12** to variable **v3**. The form **v3 = -12** is not allowed.

Hexadecimal numbers are prefixed by **0x.** The range for hex numbers is from **0x0000 to 0xFFFF**.

For example: **0x12** means **12 hex** which is 18 in decimal.

Note: Negative numbers correspond to the following hex equivalents: **0x8000 = -32768; 0x8001 =** - **32767; 0x8002 = - 32766; 0xFFFF = -1**

Variable Types:

Numeric variables have the form **v*n*** where ***n*** range from 0 to 19.

Examples:

v5 = 10 + 2 assigns the decimal value 12 to numeric variable v5.
v9 = 0x10 * 5 assigns the value 80 to numeric variable v9.

String variables have the form **s*n*** where ***n*** range from 0 to 19. The maximum length for a string is 256 characters. Literal strings must be enclosed between a pair of quotes.

Example:

s3 = "Hello" + " " + "World!" assigns the string **"Hello World!"** to string variable s3.

Note:

The following sequences in a string have special meanings:

\n	New-line character.
\r	Carriage-return character.
\t	Tab character
\	Backslash character
\e	Escape character
\x*nn*	Literal hex character.

Example: If you want to send the hex character 0x08   to the emulator, use the following string:

send "\x08"

Note: After the **\x** there **must** be two hexadecimal characters!

See also the **str** and **val** functions for formatted numbers and conversions.

Numeric Operators:

The following operators are supported and are listed from the highest to the lowest precedence:

( ) **Parenthesis** alters the precedence of operators.

Example:

v5 = (1+2) * 3 assigns 9 to variable v5.

***** and **/** **Multiply (*)** and **Divide (/)**

Examples:

v1 = 2 * 3 assigns 6 to variable v1.
v2 = 7 / 2 assigns 3 to variable v2, **not 3.5** since the variables are always integer variables.

+ and - **Plus (+)** and **Minus (-)**

Example:

v4 = 1 + 2 * 3 - **4** assigns the result 3 to variable v4.

<< and **>>** **Left Shift (<<)** and **Right Shift (>>).**

Examples:

v5 = 10 << 3 assigns the result 80 to variable v5.
v6 = 0x1234 >> 4 assigns the result 0x123 to variable v6.

<, <=, >, >= **Smaller than (<), Smaller or equal (<=), Greater than (>), Greater or equal (>=)**

Example:

v2 = v4 >= 5 assigns 1 to variable v2 if variable v4 is greater than or equal to 5 and assigns 0 to variable v2 if variable v4 is smaller than 5. In this context, 1 means true, while 0 means false.

!= and **==** **Not Equal (!=) and Equal (==)**

Example:

```
if v5 != 7
    v8 = 7
endif
```

assigns 7 to variable v8 if v5 is not equal to 7.

& Bit wise AND. Performs the Boolean AND operation on bits.

Example:

v9 = v9 & 0x0F keeps only the lower 4 bits of v9 while making the upper 12 bits zero.

Note: 0 & 0 = 0
0 & 1 = 0
1 & 0 = 0
1 & 1 = 1

^ Bit wise Exclusive OR. Performs the Boolean XOR operation on bits.

Example:

v10 = v10 ^ 0xFFFF inverts all the bits of v10.

Note: 0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0

| Bit wise OR. Performs the Boolean OR operation on bits.

Example:

v11 = v11 | 0xFF00 sets the upper 8 bits of v11.

Note: 0 | 0 = 0
0 | 1 = 1
1 | 0 = 1
1 | 1 = 1

&& Logical AND. Performs the logical AND of two values.   Results in a 0 or 1 depending on the other operands.

Example:

```
if ( ( v10 == 5 ) && ( v11 == 6 ) )
        v12 = 1
endif
```

sets v12 to 1 only if v10 is equal to 5 and v11 is equal to 6.

Note: **v13 = 0x1234 & 0x00FF** assigns 0x0034 to v13.
v13 = 0x1234 && 0x00FF assigns 0x0001 to v13.
v13 = 0x1234 && 0x1200 & 0x00FF assigns 0x0000 to v13.

|| Logical OR. Performs the logical OR of two values. Results in 1 when either operand is not zero.

Example:

```
if v14 < 12 || v14 > 14
        v15 = 1
endif
```

sets v15 to 1 only if v14 is not equal to 13.

Note: **v15 = 0x1234 | 0x00FF** assigns 0x12FF to v15.
v15 = 0x1234 || 0x00FF assigns 0x0001 to v15.
v15 = 0x1234 || 0x1200 | 0x1234 assigns 0x0001 to v15.

Assignment:

<numeric variable> = <numeric expression>

Example:

```
v2 = 10 * v1
```

If v1 has the value of 5, v2 is assigned the value 15

<string variable> = <string expression>

Example:

```
s1 = s0 + "world!"
```

If s0 has the contents "Hello ", s1 is assigned the contents "Hello world!"

Program Flow Control:

if, else and **endif:**

```
if <expression>
        If <expression> evaluates to non-zero, then
        this code is executed until a matching else or endif is found,
else
        but when <expression> evaluates to 0
        this code is executed until a matching endif is found.
endif
```

Notes:

if can be nested to a maximum of 20 levels.

<expression> must be a numeric expression

Example:

```
if v14 > v15
        if v16 == v17
                display "v14 > v15, v16 == v17"
        else
                display "v14 > v15, v16 != v17"
        endif
else
        if v11 <= v12
                display "v14 <= v15 and v11 <= v12"
        endif
endif
```

Example: To test for the password prompt on the line of the cursor:

```
s1 = screen(0,row,79)
if instr(s1,"sword")+1
        send "my_password"
else
        // Error condition: pop up debug window
        display "Error: No password prompt!"
        display "Cursor is at "+row+","+col
        display s1
endif
```

while, endwhile:

```
while
                While <expression> evaluates to non-zero, execute this code until an
                endwhile statement is found.
endwhile
```

Notes:

while can be nested to a maximum of 20 levels

<expression> must be a numeric expression

Example:

```
v2 = 3
while v2 != 0
        display "v2 is not zero"
        v2 = v2 - 1
```

endwhile

File Functions:

open <string expression> <file handle> [<open_mode>]

Example:
open "c:\autoexec.bat" 2 opens the file "c:\autoexec.bat" and assign it to file handle **2**.

<open_mode> is the optional file mode.   Specify as a string (with quotes) as follows:

"rt"	read text (default)
"wt"	write text
"at"	append text

Notes:
File handles can range from 0 to 3. This means you can work with up to 4 files.
If you specify a wild card filename like "*.sys", a dialogue box prompts you to select a file and if <string expression> is a string variable, it is assigned the selected file name.
If <open_mode> is not specified, it is taken as "rt"
Variable **ferr** is set according to the result of the **open** that is 0 if OK, non-zero if not.
An **ferr** of 2 indicates file not found for "rt" mode.

Example:
s10 = "c:\address\*.dat"
open s10 0 "at"
s11 = "This is the end of file: " + s10
writeln 0 s11
close 0
The **open** statement will prompt the user for the file to be opened and the comment in s11 will be appended to the file.

close <file handle>

Example:
close 2 closes a previously opened file on handle **2**.

readln <file handle>
Function which reads a line from the text file specified by <file handled> previously opened by **open**.   The CR-LF at the end of lines are converted to a single LF character.
Example:
s19 = readln 2 reads a line of text from the previously opened file using handle 2 into string variable s19.

Notes:
<file handle> is a numeric expression.
The variable **ferr** is set according to the result of the read that is 0 if OK and non-zero if not.   An **ferr** of -1 indicates the end of the file.
Example:
v10 = 3
open "*.sys" v10
s11 = readln v10
close v10
Read the first line from the selected .sys file into string variable s11 using file handle 3.

writeln <file handle> <string expression>

Writes <string expression> to text file specified by <file handled> previously opened in "wt" or "at" mode. The variable **ferr** is set according to the result of the read that is 0 if OK and non-zero if not.

Note that the "\r\n" string should be used to write text onto new lines for DOS text files.

ferr

This function gives 0 if last file operation was OK and non-zero if not.

Example:

```
open "file.scr" 0
v12 = ferr
```

assigns 0 to v12 if file.scr exists, or non-zero value if it doesn't

Input and Output Functions:

display <string expression> [<nexp0>] ... [<nexp7>]

The string expression is displayed in the script debug window in one line. The next display command displays text in the next line. Up to 8 numeric expressions can be specified with corresponding formatting characters in <string expression> (see the **str** function for an explanation).

Note: Though the **display** command and script debug window are designed for debug purposes, it may be used in production scripts. The major side effect is that the keyboard interrupt for the **wait** command is disabled when the debug window is the active window. Any keystroke will break the script to give access to the debug menu. You need to switch back to the emulator window to interrupt a wait. Rather try to use the **send** command to echo user instructions from the server as a comment or message in the server application.

screen(<column>,<row>,<length>)

A function which reads a string of <length> characters from the screen starting at <column> and <row>. Note that <column> and <row> start at zero. The functions **row** and **col** give the cursor position.

Example:

```
s1 = " "
while s1 != "Password?"
wait 5
s1 = screen(0,row,9)
endwhile
```

Assigns "Password?" to s1 if the row containing the cursor starts with "Password?".

show <expression>

To stop display on screen for aesthetic reasons and restart it, use **show**. If <expression> evaluates to non-zero, the display screen is updated with new data, else it is not updated. When the script terminates, screen updating resumes if it was turned off.

send <string expression>

The string expression is sent to the emulator as if the data was typed from the keyboard.

wait <ticks> [<event mask>]

The **wait** command waits for the specified number of ticks (where one tick = 1/18 seconds) for the specified event or events. The events with their respective bit mask are:

1 - The number of ticks elapsed.
2 - Keyboard input was received.
4 - Data was received from the host.

To enable an event, the corresponding bit must be set in the <event mask>. To enable all the events, an event mask of 7 is used. If no <event mask> is specified, 7 is used by default.

The **flags** function can be used to determine which event caused the **wait** to terminate. The value of **flags** will be 1, 2 or 4 after termination of **wait** according to the terminating event.

Note: The keyboard interrupt for the **wait** command is disabled when the debug window is the active window as caused by the **display** or **debug** commands. You need to switch back to the emulator window to interrupt a wait.

Example:
wait 18
This will wait for 1 second if no keyboard or host activity is detected in that time.

Example:
wait 1 6
This will wait for an indefinite period if no keyboard or host activity is detected.

Example:
send "Press almost any key to continue..."
wait 1 2
This is equivalent to many languages' pause instructions with a suitable server application that would echo the instruction to the emulator screen.

flags

Function returning the reason for termination of a **wait** as a bit map: Time-out 1, Keyboard 2, Host receive 4.

Example:
```
        wait 20
        v1 = flags
```
Gives 1 if nothing happened for 20 ticks, 2 if the user typed a key while waiting and 4 if data was received from the host while waiting.

Example:
```
wait 36
if flags & 1
display "Timed out!"
else
if flags & 2
        display "User break!"
else
        display "Host responded"
endif
endif
```
This will wait for up to 2 seconds and reports what caused the end of the wait: time-out, keyboard or host activity.

Example:
To wait for a prompt: Wait for a delay before transmission and then wait for the transmission to complete before continuing the script.
```
wait 180      // Wait up to 10 seconds
v12 = 4              // 4 means data was received.
while v12 > 3        // Wait for time-out or user break
wait 1               // Wait for more data
v12 = flags   // (time=1, key=2, data=4).
endwhile
```

key

Function returning the key that was last pressed as a string.

For example:

If you press the "a" key, the **key** function results in "a". But if you press a function key, the programmed string is the result. Suppose **Alt-F1** was programmed as "Hello World!". The **key** function gives the result "Hello World!" when you press **Alt-F1**.

Note: To use the **key** function, you must precede it with the **wait** function. If there was no **wait** function before **key**, the string from the last key pressed, is given.

Example:

```
wait 18
if flags & 2
        send key
endif
```

Wait for a keystroke and send it unchanged to the emulator.

col

This function gives the column position of the cursor on the screen.
NOTE: the first column is taken as 0, not 1.

Example: If the cursor is at column 12,

```
      v10 = col
```

assigns 11 to v10.

row

This function gives the row position of the cursor on the screen.
NOTE: the first row is taken as 0, not 1.

Example:
If the cursor is at row 13,

```
      v11 = row
```

assigns 12 to v11

String Functions:

All the functions resulting in a string can be used in string expressions for further manipulation. Functions resulting in numeric results can be used in numeric expressions.

+ The + operator concatenates two strings.

Example:
s10 = s11 + " seconds" assigns "six seconds" to s10 if s11 contains the string "six"

arg(<number>)

The string expression **arg**(<number>) takes the value of the relevant command line argument AFTER parsing. That is the "/*option***=**" parts are stripped.

Example: If you specify
iw3270.exe /=+Mainframe+/s=Netscape.scr
In this example, arg(0) is "/="
arg(1) is "Mainframe"
arg(2) is "netscape.scr"
arg(n) is ""

mid(<string>,<first>,<last>)

Gives a sub-section of a string.<first> and <last> are numeric expressions specifying the first and last characters. Note that the first character of a string is at 0 and **not at 1.**

Examples:
display mid("abcdefg", 2, 4) displays "cde"
display mid(s10, len(s10) - 2, len(s10) - 1) displays the last two characters of string variable s10

str(<format>,val1,val2,val3 ...)

Gives a C-language formatted string built up from **numeric** variable arguments.

Example:
display str("Signed value = %d, unsigned value = %u", v11, v11) displays:
Signed value = -1, unsigned value = 65535
when v11 = 0xFFFF

The following formatting characters can be used in the format string:

%d	a signed decimal value, like -6
%u	a unsigned decimal value, like 65530
%x	a hex number in lower case, like 12ab
%X	a hex number in upper case, like 12AB
%c	ASCII value

Using unsupported formatting characters like %s or %f will lead to unpredictable results.

val(<format>,<string>)

Returns the numeric value of <string> converted using <format> (using supported C conventions)

<format> can be one of the following:

%d	signed decimal value
%u	unsigned decimal value

%x	hex value
%o	octal value
%i	decimal, octal or hex value
%c	ASCII value of first character

xlate(<string>,<table>)

Translate all characters in <string> according to <table>
<table> is a string expression containing replacement characters in order of the ASCII values of the characters to be replaced. To perform a translation of all characters, a table of 256 characters must be given. Note that the first code corresponding to a value of zero is never used, because a value of zero (null) is used to indicate the end of a string. If only control codes need to be translated, the table needs to be only 32 characters long. All characters having values over 31 will not be translated in this case.

Example:
To translate all control characters except tab (9), line-feed (10) and CR (13) in s4 to '*':
s4 = xlate (s4,"*******\t\n**\r******************")**

find(<string1>,<string2>)

Finds the first occurrence of ANY character from <string2> in <string1> and returns its position (starting at 0).
If no character from <string2> is found in <string2>, a value of   -1 is given.

Example:

s1 = "performance"
s2 = "mf"
v3 = find(s1, s2)

v3 is assigned the value 3

instr(<string1>,<string2>)

Finds the first occurrence of <string2> in <string1> and returns its position (starting at 0).
If <string2> is not found in <string1>, a value of -1 is returned.

Example:

s1 = "performance"
s2 = "man"
v3 = instr(s1, s2)

v3 is assigned the value 6

len(<string>)

Gives the length of a string.

Example:

v1 = len("Inet Software")

v1 is assigned the value 13.

match(<string1>,<string2>)

Compares <string 1> to <string2>. Returns 1 if match, else 0.

Example:

v1 = match("software", "soft")

Gives 0, since the two strings are not the same in content and length.

Start Other Programs:

execute <string expression> [<window mode>]

Executes the program specified by <string expression>

Example:

execute "c:\winet\iwftp.exe /h=RemServ"

<window mode> is an optional value specifying the way the window of the program to be executed must be started:

0 Hide
1 Normal
2 Minimised
3 Maximised
4 Not Activate
5 Show
6 Minimise
7 Min Not Active
8 N/A
9 Restore

Debug Options:

debug

Breaks the script execution, activates the debug mode and pops up the script debug window.   Any illegal command will also start the debug mode.   When the script debug window has been opened by the **display** command, any key can be used to break the script into debug mode.

The following interactive debug commands are available by pressing the required key:
T - Trace step by step through the script
G - Go fast again
E - Expression trace toggle - display evaluations at each step
V - Variables are displayed - all the numeric variables
S - String variables are displayed - all the non-zero ones
Other keys - Cause break or display debug menu

Note: The keyboard interrupt for the **wait** command is disabled when the debug window is the active window as caused by the **display** or **debug** commands.   You need to switch back to the emulator window to interrupt a wait.

Example Scripts:

A few general purpose scripts are given to demonstrate auto-logon and up- or download are included as examples on disk. Look for *.SCR files in your WINET directory. See the Command Line Options on customising icons with scripts.

A quick and dirty script to login to a typical Unix server. You might have to change the time-outs.

```
wait 90                        // Wait for: 5 seconds, any data from server or user break.
v12 = 4                        // Loop to wait for prompt. 4 means data was received.
while v12 > 3                  // Stay in this loop until a time-out(1) or user break(2)
occurs.
      wait 3                   // Wait 3 ticks for more data to be received.
      v12 = flags              // Wait terminated by: (time=1, key=2, data=4).
endwhile
send "iamme\n\r"               // Type user id.
wait 18                        // Wait 1 second for next prompt to arrive.
send "itsasecret\n\r"          // Type password.
```

Example intelligent script to logon to a typical IBM3270 server. This is a more rugged script to cope with delays and minor variations in the application software layout.
When testing, use debug to break and pop-up the script window where you may examine variables and trace step-by-step through the script. (Note that the script window then takes control of the keyboard. The emulator window must be activated to terminate a wait with a key stroke where needed.)

```
//debug                 // Stops the script to allow debug commands.
wait 90                 // Wait for: 5 seconds, any data from server or user break.
v11 = flags             // Wait flag (time out=1, key pressed=2, data received=4).
                        // v11 is used in an if-else-if construction to handle
                        // and report on the above three cases.
if v11 & 4              // Did receive data from host.
   v1 = 10              // The number of times checking for login prompt
                        // to take care of delays in the network transmissions.
   while v1 > 0
// When a prompt is part of a whole new screen or many lines, use this
// construction to wait for the end of the transmission.
      v12 = 4           // Loop to wait for prompt. 4 means data was received.
      while v12 > 3     // Stay in this loop until a time-out(1) or user break(2) occurs.
         wait 1         // Wait one tick for more data to be received.
         v12 = flags    // Wait terminated by: (time=1, key=2, data=4).
      endwhile
      s1 = screen(0,row+1,77)     // Copy line of the cursor to string variable.
      if instr(s1,"Userid")+1     // Did it contain the Userid prompt?
         send "my-user-id\x09"    // Type user id and tab to next field.
         send "my-password\r"     // Type password and transmit.
         v1 = 0                   // Stop while loop successful.
      endif
      v1 = v1 - 1
   endwhile                       // Next test for login prompt.
   if v1 == 0           // The while loop waiting for the login prompt
                        // ran out without success.
      display "Time out!  Did not received login prompt after v1 pauses in
transmission."
      display str("The cursor is at row %d and column %d. Last line read was: ",
row, col)
      display "<" + s1 + ">"
//    debug             // Stops the script to allow debug commands.
   endif
```

```
else
if v11 & 2              // Initial wait terminated by key.
   s11 = key
   display "User aborted before any data was received from server!"
   display "Key pressed was " + s11
else
if v11 & 1              // Initial wait terminated due to time out.
   display "Received no reply from server.  Connection down?"
endif
endif   //else if
endif   //else if
```

Sample of a possible method to download data from a VT100 session. This method reads an 80 column line from the screen every time the cursor moved left. On slow machines it may happen that lines are lost. The CR characters can not be read from the screen. All lines are padded with spaces up to column 79.
A better method is to use the **Write** menu option in the emulator.

```
open "*.*" 0 "wt"             // Prompt the user for a file name.
//open "download.dat" 0 "wt"  // Open a file for writing.
if ferr
    display "Error opening file"
endif
send "cat download.file\r"    // The command to display file to screen.
wait 4
v0 = 0
while flags & 4
    v1 = col
    if v0 > v1                // The cursor moved left, meaning a new line
        s0 = screen(0,row-1,79)   // Read previous line
        writeln 0 s0 + "\n"   // Write it, adding the LF needed in DOS files
    endif
    v0 = v1
    wait 4
endwhile
close 0
```

The following script demonstrates loops, arithmetic and output options to display a multiplication table.

```
display "Multiplication tables"
display "---------------------"
v1 = 1                    // set row value to 1
v2 = 1                    // set column value to 1
s0 = "%5u"                // format to display each number
s1 = ""                   // build row in string variable 1
while v1 <= 12            // repeat 12 times
  while v2 < 13 // also repeat 12 times!
      v0 = v1 * v2        // multiply!
      s1 = s1 + str(s0,v0)      // build up row
      v2 = v2 + 1         // increment column
  endwhile                // end of column loop
  display s1              // display the complete row
  s1 = ""                 // clear row string
  v2 = 1                  // start a new row
  v1 = v1 + 1             // increment row
  endwhile                // end of row loop
```

Here are some shortcuts to:

Number Formats
Variable Types

Numeric Operators
String Functions

Override Control with the .INI Files

If you change some of your settings and save it to your .INI file but the setting seems never to be loaded from the .INI file, you may be trying to change a setting that has been locked by the system manager.   You can check in the **Help/About** box if an override file has been used.

Inet for Windows has an override facility to allow precise and selectable control by Network Managers on a server based LAN and still allow users to modify parts of their own configuration.   The UTS60 shares this facility and it can be activated by a system manager for the UTS60 emulator without using any of the other **Inet** products.

The Network Manager may lock certain features by specifying them in files in the WINET directory where users do not have write access.   This feature is most often used to define variations in the terminal emulators.   Different levels of defaults may also be set.

Users may still change all settings for their current sessions and save them to their own INET.INI files, but reloading **Inet for Windows** will override any locked options.

To enhance the security of this system, the instructions on how to activate the override is contained in a seperate file, OVERRIDE.WRI, which should not be available to general users. The sequence in which **Inet** loads .INI files until it finds the required information is also outlined there.

Frequently Asked Questions

In this section, frequently asked questions will be added with every new release.

Printing doesn't work!

You click on 'Open' in the emulator and nothing happens?

My settings are not saved!

The emulation seems to be faulty!

Error message: what does it mean?

DONTWIPE.ME! : what and where is it?

See also: Open, Close, Setup Menu, UTS Settings

How do I find the problem in my printing?

Printing in the Windows environment can be quite confusing. I am not going to try explaining it, but rather give a few steps to test.

1. Try to print from a simple Windows application like NotePad. Carefully check that the correct printer has been selected with the **Print Setup** option. If this does not work, do NOT try to get the **Inet** printing to work - the problem lies with your Windows installation, physical printer, printer driver or network print queue configuration. You must have the correct printer driver for your printer.

2. The **Inet** print servers may use a spool directory (depending on the way they have been set up). The environment variables, INETSPOOL, TEMP or TMP are used to define the spool directory. The first defined variable is used to define the spooling directory. If it points to a non-existing directory or a full or write protected drive, printing will fail.

3. The LPD print server always uses the spool directory - if the spool directory is empty, either the files have been printed or no print jobs have been received.

What is wrong if you click on OPEN and nothing seems to happen?

This happens when the remote server specified can not be accessed.   If no error message appears, check that the server can be reached.

You must be able to **ping** the remote host before trying to connect to it.   Ping is a function of the WinSock you are using.   The **Inet for Windows** full product includes an **Inet Ping** icon that you should use.   With MS TCP/IP under early versions Windows 95 you have to use the command line (DOS like) ping program or a third party **ping**.   If you can not ping the remote, it may be a problem on the network or your WinSock interface and routes may be set incorrectly.

Check the bottom status bar for any messages.   If it says "Connection refused" or "Session closed", then it means that the remote server did reply, but in a negative way.   You may have the wrong IP address or the wrong TCP port number.   Check that the address and port are correct and correctly configured at the remote host.

My settings are not saved!

There are 3 things to check:
1. In the terminal emulators you must select the **Setup/Save** option to save new settings to the .INI file.
2. Some command line options might override your settings.
3. The system administrator might have defined an override file that may lock selected settings.   Check under **Help/About** for an override file.

The emulation seems to be wrong!

If you suspect a bug in the way **Inet** emulate your terminal type, please check a few things first.   Some apparent emulation problems are due to incorrect terminal settings.   Check all your selections in Setup/Settings.   If text does not display or appears in the wrong colour, see the Colour and Palette setup items.   Keys that do not perform their proper functions may easily be reprogrammed from Setup/Function Keys.

If you are certain that the emulation is faulty, please provide us with a log of the problem: Make sure that **Write Binary** is selected in Setup/Settings. Before starting the session, start a Write to a log file. Now produce the faulty screen in the least number of steps and as soon as you have done it, close the log file by activating Write again and closing the file. Please compress (zip) and e-mail the file to us with a description of the problem and what you consider wrong with the emulation - even if it seems obvious to you. You know your application, we may never have seen it. We will take prompt action to identify and rectify the problem.

See our **E-mail address**

Error messages confuse me!

It is important to distinguish where an error message comes from.   All error messages which appear as text in the emulator screen, come from the remote host or mainframe.   You should speak to the mainframe manager.   In the emulator, you must check and verify the IP address and TCP port on the Open selection and the selections in Setup/Settings.

Messages in pop-up boxes are generated by software on your PC.   Look at the header of the box to determine the offended program.   However, the error may be with some other program.   Important things to check are your TCP/IP implementation and configuration. Test your connections using a ping program to make sure the network is working.   We won't be able to help you solve network related problems or to configure your TCP/IP services.

If you have tried everything and it is a repeatable problem on other PC's, please describe the problem in detail and e-mail us.

See our **E-mail address**

DONTWIPE.ME! is your license file

The WINET environment variable is used to find the licence file, DONTWIPE.ME!, and initial setup files such as the .INI files.   The license determines which **Inet** products and how many copies may run at the same time.   The license information is displayed in the **Help/About** box.

During installation, an additional copy of the DONTWIPE.ME! file is copied to the \INET directory in the root of the current drive, to enable you to use the emulator before setting the WINET variable.   This copy can be removed once WINET is set.   If you have copied your installation to another place, you should change the WINET environment variable in your AUTOEXEC.BAT to point to the new directory.

What is Inet UTS60?

Inet UTS60 is a Sperry/Unisys UTS60 terminal emulator for Windows providing access to Sperry/Unisys 1100/2200 mainframes using TCP/IP. It runs on any WinSock 1.1 compliant TCP/IP stack under the Windows 3.1x, Windows 95 or Windows NT operating systems.

It supports the following:

- DCP and HLC protocols.
  - Demand and TIP environments including Mapper.
  - Easy key mapping with over 200 function keys.
  - Install program with configurable install script.
  - Full script language including special functions like:
  - File read and write.
- Execute external DOS/Windows programs.
- String, numeric and bit-wise operations.
- Addressable cursor and screen reads.
- Keyboard simulation.
- Colour, font and screen size selection.
- Screen printing and screen-bypass printing to printer or file.
- Multiple sessions can be established.
- Mouse support: copy-and-paste in emulator and to/from other Windows applications.
- Full settings override control by system managers.

What is Inet TCP/IP?

(This general information along with an overview and full specifications are given in the INETINFO.WRI file.)

Inet TCP/IP is an easy-to-use professional Internet Protocol package for PCs running DOS or Windows, which provides a powerful data communication facility whereby network users can communicate with other computers and their printers; including PCs, minis and mainframes. **Inet TCP/IP** adheres to international standards for various terminal emulators, TCP/IP, Telnet and FTP.

Inet TCP/IP is a networking communications application providing both client and server services on the network, which allows you to:

Run applications on a TCP/IP host system (like a Unix box or a main frame) from your workstation.
Transfer data between your workstation and TCP/IP hosts.
Run network aware applications on your workstation
Print to your workstation from TCP/IP hosts (and vice versa)
Run a low cost Router or Terminal Server

The full **Inet** TCP/IP package consists of **Inet for Windows**, **Inet for DOS** and **RINET**. To suit specific needs, **Inet for Windows** is sold in 4 different configurations and some emulators are sold as separate products. Though **Inet** TCP/IP contains a power house of tools, it is priced to suit users needing only a few of the facilities provided.

Main features include:

- Telnet terminal emulations with configurable keyboards, colour, 132 column, scripts, etc.
- File transfer protocol servers and clients (FTP)
- Electronic mail (SMTP)
- Network printing (socket printing, LPR/LPD and 3287 for TN3270E)
- Modem support (SLIP and CSLIP)
- Built in network monitoring and tracing (LanScope also available as a separate product)
- Network IP routing, gateways, async server, etc.
- Winsock 1.1 compatibility under Windows

Inet will run on ODI, IPX, NDIS, NetBios or packet driver interfaces. Alternative roles of **Inet** include that of IP router (internal, X.25 or async) or terminal server.

Inet will co-exist with Novell, Lan Manager, Windows for Workgroups, Windows 95, WinNT and other networking operating systems by sharing network adapters.

The main feature of **Inet** is its comprehensive terminal emulation including VT100/220/300, IBM3270/8/9/E, D211, HP700/92, Sperry UTS60, Burroughs ET1100 and Wyse50. **Inet for DOS** also supports any third party terminal emulator which uses the Interrupt 14 (INT14) interface.

The Windows version of **Inet** includes features like:

- Cut-and-paste operation between local and remote system
- Sizing and scrolling of terminal emulation windows
- Full script language
- Winsock 1.1 API

- Drag-and-drop operation for file transfer and network printing.

RINET is a resident (TSR) version of the **Inet for DOS** protocol stack. RINET provides a built-in socket print server and a transparent socket printing redirector. It also provides RFC compliant Netbios.

Inet is distributed and supported in South Africa by Lan Design, a member of the Altech Data group.

NB: The full specifications are given in the INETINFO.WRI file. Please consult it for information on the latest Inet products.

The developers

GP van Niekerk Ondernemings is a software company, specialising in terminal emulators, TCP/IP and printer support for IP based networks. The head office is in Pretoria, South Africa. We have been in this business for eleven years. We are proud of our fast turn-around time for fixing bugs and releasing new updates. We can supply tailor made implementations to our customers on short notice.

More than 40 000 copies of our products have been sold in South Africa mostly to large companies and state departments.

E-mail address

Back in 1997 we could be contacted at: Ferdi@osi.co.za or GerritvN@osi.co.za
You can contact us for news on current developments and updates. We prefer that support should be done by your supplier. It is possible to negotiate a support contract with us for direct support.