WINVOKE

Version 2.0

winvoke.exe takes command line arguments where the first argument
is the name of a file that winvoke will read for command input
using a very simple winvoke syntax.

Command Line Substitutions

Within the file containing winvoke commands, the following
substitutions are made:

 {CurDir} = Current Windows Default Directory
 {UserName} = EINet User Name (only if LOGIN command was used
              prior to referencing this variable)

Winvoke argument substitutions:
 {arg0} = argument 0 from winvoke command line
 {rest0} = all arguments past argument 0 from winvoke command line
 {arg1} ...
 {rest1} ... and so on

Comment Lines

winvoke accepts the following characters as comment delimiters when placed
as the first character of any line:

    : (colon)
    ; (semicolon)
    - (minus)
    + (plus)
    ' (single quote)
    | (verticle bar)

Commands

winvoke accepts the following commands:

TITLE text

This command gives the executing WINVOKE task the given title text. This title is both as the Windows task name and as the title of any user dialog boxes.

LOGIN

This command initiates EINet User Authentication (Login) if the user is not presently authenticated. If the user cancels the login dialog, winvoke stops processing commands. (This command is not supported in EINet shareware and freeware releases.)

LOGIN&

This command operates in the same manner as LOGIN except it does not wait for the user to be successfully authenticated. (This command is not supported in EINet shareware and freeware releases.)

INVOKE `filename ...`

This command invokes an application using the rest of the command arguments on the line. winvoke then waits until the application terminates before processing the next command.

INVOKE& `filename ...`

Same operation as INVOKE but it does not wait for the application to terminate.

VIEW `filename ...`

This command invokes the associated Windows application for viewing the named file using the rest of the command arguments on the line. winvoke then waits until the application terminates before processing the next command.

VIEW& `filename ...`

Same operation as VIEW but it does not wait for the application to terminate.

SENDKEYS `...`

This command sends the key sequence specified on the command line to the application just invoked by the previous INVOKE& or VIEW& command. See the appendix below for the Microsoft documentation on the sendkey syntax. (Note that the Appendix is copyright(c) Microsoft.)

WAIT

This command waits for the previously application launched with INVOKE& or VIEW& to terminate.

EXPAND `fromfile tofile`

This command expands the designated LZH-compressed file.

DELETE `filename`

This command deletes the specified file.

MSGBOX {message}
This commands displays a message box dialog to the user.

Example Commands:

```
MsgBox First arg is: {arg0}
MsgBox Rest of args are: {rest0}
Login
Invoke notepad c:\windows\win.ini
View& c:\windows\progman.hlp
SendKeys %sorganizing documents%s%g
Wait
MsgBox aaa {UserName} is done.
Delete c:\xxx.asc
```

APPENDIX

SendKey Documentation
Copyright (C) Microsoft

Each key is represented by one or more characters. To specify a single keyboard character, use the character itself. For example, to represent the letter A, use "A" for keytext. If you want to represent more than one character, append each additional character to the one preceding it. To represent the letters A, B, and C, use "ABC" for keytext.

The plus sign (+), caret (^), percent sign (%), tilde (~), and parentheses ( ) have special meanings to SendKeys. To specify one of these characters, enclose it inside braces. For example, to specify the plus sign, use {+}. Brackets ([ ]) have no special meaning to SendKeys, but you must enclose them in braces as well, because in other applications for Microsoft Windows, brackets do have special meaning that may be significant when dynamic data exchange (DDE) occurs. To send brace characters, use {{} and {}}.
To specify characters that aren't displayed when you press a key (such as Enter or Tab) and keys that represent actions rather than characters, use the codes shown below:

Key	Code	Key	Code
Backspace	{BACKSPACE} or {BS} or {BKSP}	Break	{BREAK}
Caps Lock	{CAPSLOCK}	Clear	{CLEAR}
Del	{DELETE} or {DEL}	Down Arrow	{DOWN}
End	{END}	Enter	{ENTER} or ~
Esc	{ESCAPE} or {ESC}	Help	{HELP}
Home	{HOME}	Ins	{INSERT}
Left Arrow	{LEFT}	Num Lock	{NUMLOCK}
Page Down	{PGDN}	Page Up	{PGUP}
Print Screen	{PRTSC}	Right Arrow	{RIGHT}
Scroll Lock	{SCROLLLOCK}	Tab	{TAB}
Up Arrow	{UP}	F1	{F1}

F2	{F2}	F3	{F3}
F4	{F4}	F5	{F5}
F6	{F6}	F7	{F7}
F8	{F8}	F9	{F9}
F10	{F10}	F11	{F11}
F12	{F12}	F13	{F13}
F14	{F14}	F15	{F15}
F16	{F16}		

To specify keys combined with any combination of Shift, Ctrl, and Alt keys, precede the regular key code with one or more of the following codes:

Key	Code
Shift	+
Control	^
Alt	%

To specify that Shift, Ctrl, and/or Alt should be held down while several other keys are pressed, enclose the keys' code in parentheses. For example, to have the Shift key held down while E and C are pressed, use "+(EC)". To have Shift held down while E is pressed, followed by C being pressed without Shift, use "+EC".
To specify repeating keys, use the form {key number}; you must put a space between key and number. For example, {LEFT 42} means press the Left Arrow key 42 times; {h 10} means press h 10 times.