Built-in Commands

Overview

- Powerpro's built-in commands allow manipulation of running programs & windows, plus control of the Windows interface.

General Syntax

- **Command.Action(*arg(1), arg(2), ..., arg(n)*)**

Commands

Bar	Bars are groups of buttons, created either as a command list in the PPConfig dialog or with scripts. Bars can be manipulated, altered or repositioned using the Bar actions.
Clip	The Clip command is used to work with the clipboard, with actions for manipulating, tracking & editing of the system clipboard.
Configure	Allows import & export of PowerPro configuration files.
Desktop	The Desktop command is used to control various aspects of the Windows desktop layout.
Exec	The Exec command includes various functions for file, window, system, multimedia, & PowerPro manipulations.
File	The File command accesses PowerPro's internal file manipulation commands for manipulating files.
Format	Format commands are used for formatting bars & menus. Many of these will only work in particular situations, as specified.
Keys	The old format *Keys command becomes Win.SendKeys() with PowerPro's new expression syntax, & there is also the equivalent command Win.Keys(), which uses a different behind-the-scenes method to send the keys. This Keys section of PPSR provides a reference for the target window & keyboard key codes used with both Keys functions.

Macro	Keyboard macros let you create predefined text shortcuts that, when typed, are replace by the full text.
Menu	The Menu command is used to display a menu.
Message	The old format *Message command is replaced by the built-in function messagebox() in the new expression syntax.
Mouse	The old format *Mouse command becomes Win.Mouse() with PowerPro's new expression syntax.
Note	The Note command lets you work with PowerPro notes.
ScreenSaver	The ScreenSaver command is used to work with the Windows screen saver.
Script	The old format Script command encompassed a range of functions relating to control & manipulation of PowerPro scripts. Most of these commands are now deprecated for scripting use, and the rest are all mentioned in other places of this reference.
Shutdown	The Shutdown command has functions to shutdown/exit both PowerPro and Windows.
Timer	Powerpro has 26 customisable timers, identified by the single-letter labels a, b, c, ..., z.
TrayIcon	The *TrayIcon command is used to simulate mouse clicks on tray icons from any program.
Vdesk	The Vdesk command works with PowerPro's virtual desktops feature.
Wait	The Wait command is used to wait for some condition before continuing. The condition can be an amount of time, an expression, or both, depending on the command & action used. Waits should always be used after commands which act on other windows, as after PowerPro sends the window a message asking it to do something, the window itself must have time to respond.

Wallpaper	The Wallpaper command is used to change the desktop wallpaper.
Window	The Window commands allow manipulation of the windows of running programs.

Notes

- Script & Format commands do not use the new Expression Syntax given in the General Syntax section above.

PrevUp Next

Bar

Overview

- Bars are groups of buttons, created either as a command list in the PPConfig dialog or with scripts. Bars can be manipulated, altered or repositioned using the Bar actions.

Actions

Close	Closes a bar and removes it from memory.	
	Format	Formats a bar.
Hide	Hides a bar but keeps it in memory (for faster reshow).	
	HideShow	Hides a bar if visible; shows it otherwise.
Keys	Readies bar to receive keys.	
	SelectSubBar	Shows the specified subbar on a PowerPro bar, hiding any other subbars.
SelectSubBarToButton	Shows the specified subbar aligned to the bar button pressed most recently.	
SelectSubBarToMouse	Shows the specified subbar aligned to the mouse.	
	Show	Shows a bar, optionally using slide animation to do so.
ToMouse	Temporarily moves the specified bar to the mouse, showing it if hidden.	

PPSR | *Built-in Commands* | *Clip*

Clip

Overview

- The Clip command is used to work with the clipboard, with actions for manipulating, tracking & editing of the system clipboard.

Actions

Capture	Sets PowerPro's clipboard history tracking state.
ClearClip	Clears the clip board.
ClearRecent	Clear PowerPro's list of recently captured clips.
Copy	Copies the selected text by sending Ctrl-C to the foreground window.
CopyThenToFile	Send Ctrl-C to copy the selected text, then copies the clipboard contents to a file.
Cut	Cuts the selected text by sending Ctrl-X to the foreground window.
Delete	Shows the clipboard history list, then deletes the entry selected by the user.
File	Copies a file to the clipboard.
FilePaste	Copies a file to the clipboard, then pastes it by sending Ctrl-V to the foreground window.
FileIPaste	Copies a file to the clipboard, then pastes it by sending Ctrl-Ins to the foreground window.
LongDate	Puts the date onto the clipboard in the system's long date format.
Menu	Shows a menu of recently captured clips (or, optionally, a menu of clips from another folder), allowing user to select an entry to put on the clipboard.
MenuPaste	Shows a menu of recently captured clips (or, optionally, a menu of clips from another folder), & the selected entry is

	put onto the clipboard then pasted into the active window using Ctrl-V.
MenuIPaste	Shows a menu of recently captured clips (or, optionally, a menu of clips from another folder), & the selected entry is put onto the clipboard then pasted into the active window using Ctrl-Ins.
Reattach	Puts PowerPro at the front of the queue of applications tracking clipboard history.
Paste	Sends Ctrl-V to the foreground window to paste the clipboard's contents.
ShortDate	Puts the date onto the clipboard in the system's short date format.
Text	Puts specified text onto the clipboard.
	TextAppend — Appends specified text to the clipboard.
TextPaste	Put specified text onto the clipboard, and then pastes it to the active window with Ctrl-V.
Time	Puts the current time onto the clipboard.
	ToFile — Copies the clipboard's contents to the specified text file.
ToFileAppend	Appends the clipboard's contents to the specified text file.
	Write — Sets the state of writing of clips to memory.

See Also

- For more information on PowerPro's clipboard tracking features, see:
  - *PowerPro Help CHM > Built-in Commands > *Clip*
  - *PowerPro Help CHM > Configuring with the GUI Dialogs > Clip Filters*
  - *PowerPro Help CHM > Configuring with the GUI Dialogs > Clipboard History Tracking*
- Most of the functions of the built-in Clip command work only with the first line of the clipboard. For information on working with multi-line clipboard strings, see:

- *PPSR > Plugins > The Plugins > Clip*

PrevUp Next

Configure

Overview

- Allows import & export of PowerPro configuration files.

Actions

CommandLists	Starts the PowerPro Config GUI at the CommandLists tab.	
	GUI	Starts the PowerPro Config GUI at the GUI tab.
ImportCL	Imports specified command list configuration text file into the current PowerPro configuration.	
ImportHot	Imports specified hotkeys configuration text file into the current PowerPro configuration.	
ImportSched	Imports specified scheduled events configuration text file into the current PowerPro configuration.	
Media	Starts the PowerPro Config GUI at the Media tab.	
	NewReminderMessage	Starts the PowerPro Config GUI & opens an "Edit scheduled command" dialog preset to create a scheduled message.
Scheduler	Starts the PowerPro Config GUI at the Scheduler tab.	
	Setup	Starts the PowerPro Config GUI at the Setup tab.
Timers	Starts the PowerPro Config GUI at the Timers tab.	
	WriteAllToPCF	Write the entire current PowerPro configuration, including any imported text files, to a .pcf file.

Notes

- The Key/Mouse tab cannot be opened using the new Expression syntax. Instead, use:

*Configure Key/Mouse

See Also

- For more detailed information on using .ini configuration files & the rules PowerPro uses when importing them, see:
    - *PowerPro Help CHM > Configuring with text files > Maintaining configuration using .ini files*

- For more information on configuring PowerPro, see:
    - *PowerPro Help CHM > Configuring with the GUI dialogs*

PrevUp Next

PPSR | *Built-in Commands* | *Desktop*

Desktop

Overview

- The Desktop command is used to control various aspects of the Windows desktop layout.

Actions

HideIcons	Hides all icons on the desktop.	
	HideTaskBar	Hides the Windows TaskBar.
HideShowIcons	Toggles the visibility of the desktop icons.	
	HideShowTaskBar	Toggles the visibility of the Windows TaskBar.
HideShowWindows	Toggles display of all windows on the desktop.	
	IconTextColor	Set color of text under desktop icons; use the TransIconText auto function to reset color when background changes
MinShowWindows	Toggles the minimised state of all windows on the desktop.	
RestoreIcons	Restores the relative positions of desktop icons	
	Resolution	Changes screen display resolution.
SaveIcons	Saves the relative positions of the desktop icons.	
	SaveIconsGrid	Aligns icons according to a grid & optionally also saves their relative position details to a file.
SetWorkArea	Sets the size of the maximum work area on the desktop screen.	
ShowIcons	Shows the desktop icons.	
	ShowTaskBar	Shows the Windows

	TaskBar.
ShowTaskBarAutoHide	Shows the TaskBar, then re-hides it when the mouse is moved away & the TaskBar is not the foreground window.
TransIconText	Gives desktop icon text a transparent background, optionally also setting the icon to refresh automatically after a background change.

Prev Up Next

Exec

Overview

- The Exec command includes various functions for file, window, system, multimedia, & PowerPro manipulations.

Actions

Alarms	Suspend or re-activate checks for scheduled programs.	
	Autoscroll	Starts automatic window scrolling.
AutoPress	Toggles the recognition of a window type by the mouse stop-press feature, adding it to the list of recognised window types if PowerPro does not already autopress it, or removing it if it does.	
BrowseRun	Shows a file open dialog, & the select file is immediately executed.	
CalcCalendar	Shows a PowerPro calendar dialog with 2 date/calendar fields, along with day number, week number, and the differences between the 2 dates, allowing the user to perform date calculations.	
Calendar	Shows a month-view calendar with current date highlighted, along with an input field containing the current date. The mouse or arrow keys can be used to navigate the calendar.	
CD	Controls the audio CD player.	
	ChangeConfiguration	Changes to configuration stored in a different pcf file; the new file path can be entered in the command.
ClearRecent	Clears the Windows Recent folder for the current user.	
	ClearRecentExplorer	Clears the list of recent windows shown by a Menu.Explorer call.

CommandLine	Shows a tiny command line allowing the user to enter a command that will be run, as with Windows' Start > Run dialog.
ContextMenu	Shows the right-click menu of a file, folder, or the window under the mouse. Or you can specify a file or folder to get the context menu for that file.
Disable	Disables PowerPro until the mouse is moved over a PowerPro bar, or a PowerPro hot key is pressed.
Dos	Starts Dos, runs a command line, & restarts Windows.
EmptyRecycleBin	Empties the recycle bin.
Explorer	Opens the specified folder in a single-paned Explorer window.
Explorer2	Opens the specified folder in a dual-paned Explorer window.
FindFiles	Shows the Windows Find Files dialog.
FindComputer	Shows the Windows Find Computer dialog.
HideWindow	Allows user to click a window to hide it.
HotKeys	Suspends or re-activates all or particular hotkeys.
LogKeys	Sets keystroke logging to a file.
Monitor	Suspend or re-activate repeated running of the special command list named "Monitor".
MoreCommandsAsScript	This function is only relevant in the PPConfig GUI.
Mute	Toggles the current "Mute sounds" setting.
NewFolder	Creates a new file folder.

	OnError	Sets where PowerPro error messages go and whether HookErrors list is run.
Plugin	Obsolete.	
	Print	Print a file using its associated program, such as Notepad for text files.
Prompt	Prompts for a Yes/No answer and sets a PowerPro flag according to the result.	
QuoteEscape	Sets whether single quote is an escape character in strings in expressions.	
RefreshEnvironment	Refreshes all environment variables from registry; does not delete variables which are deleted.	
RestoreLastMin	Restores the last minimized window.	
	SchedulerAdd	Adds a new scheduled event.
Scroll	Begins mouse-scrolling the window under the mouse, as if a middle-click had been pressed.	
ScrollWindow	Scrolls the window under the mouse.	
	ScrollWithMouse	Begins mouse-scrolling the window under the mouse, as if a middle-click had been pressed.
SetEnv	Set environment variable to provided text. Use the env (...) function in expressions to read environment variables.	
StandardConfiguration	Checks whether PPConfig > Setup >Advanced >"Use standard configuration" is checked & if not, outputs an error message & stops all scripts.	
Tile	Tiles the visible windows on the screen.	
	ToFile	Writes a single line of text to a log file, including a line break at the end.
VolumeAll	Set the volume for all types of playback.	

	VolumeWav	Sets the volume for .wav files. Enter number 0 (mute) to 15 (loudest). Use + or - in front of number to adjust relative to current setting.
WindowInfo	Displays/Hides a small PP window with information about the window under the mouse. Information includes the mouse screen position, plus the size, position, caption, class, and exe name of the window under the mouse.	

PrevUp Next

PPSR | Built-in Commands | File

File

Overview

- The File command accesses PowerPro's internal file manipulation commands for manipulating files.

Actions

CommandRandom	Runs a command with a randomly selected file as a parameter.	
Copy	Makes a copy of a file to the specified path.	
	CopyRandom	Copies a randomly selected file to a specified file path.
Delete	Deletes a file, or group of files.	
	DeleteNoRecycle	Completely deletes a file or group of files, bypassing the recycle bin.
DeleteOld	Deletes files in folder older than specified number of days	
	ExtChange	Change, add, or remove a file extension.
Move	Moves a file or files to another location, effectively renaming it to the new path.	
Rename	Renames a file or files, effectively moving it to the new path.	
RunRandom	Selects a file at random from a folder & runs it using the program associated with the file extension.	

See Also

- Exec.ToFile() for writing to a file
  - *PPSR > Built-in Commands > Exec > ToFile*
- The File plugin:

» *PPSR > Plugins > The Plugins > File*

PrevUp Next

Format

Overview

- Format commands are used for formatting bars & menus.
- Many of these will only work in particular situations, as specified.

Actions

BarVerticalLine	Draws a vertical line on a bar.	
	Context	Starts a menu/bar section , or shows a whole bar, depending on the active window.
ContextIf	Starts a menu/bar section , or shows a whole bar, depending on the active window, where the context is given by an expression.	
Drag	Gives click-drag functionality to any bar button assigned this command, whereby click-dragging that button will move the bar.	
EndContext	Ends a section started with a *Format Context or *Format ContextIf command.	
EndSubMenu	Ends a submenu in a menu.	
	Item	Obsolete. Use CL functions instead, see: ❍ *PPSR > CL Functions*
MaxColumn	Sets the maximum number of entries per column in a menu.	
	NewBarRow	Starts a new row in a bar.
NewBarRowLine	Starts a new row in a bar & draws a separator line.	
	NewColumn	Starts a new column in a menu.
NewColumnLine	Starts a new column with a vertical separator line.	

	Separator	Draws a vertical separator on a bar or menu.
StartSubMenu	Starts a submenu in a menu.	

PrevUp Next

Keys

Overview

- The old format *Keys command becomes Win.SendKeys() with PowerPro's new expression syntax, & there is also the equivalent command Win.Keys(), which uses a different behind-the-scenes method to send the keys.
- This Keys section of PPSR provides a reference for the target window & keyboard key codes used with both Keys functions.

General Syntax

- **Win.Keys(*"{target_window_string}keys_to_send"*)**
- **Win.SendKeys(*"{target_window_string}keys_to_send"*)**

Parameters

target_window_string

(string) keyword or caption list string indicating which window is to receive the keys, surrounded by curly braces

Optional, defaults to the active window.

In the list of Possible Values below, each "to" may be replaced by "toany":

to	works with visible windows only leaves the focus at the window receiving the keys
toany	works with both hidden and visible windows returns the focus to the window which had it before keys were sent

Possible Values

*{to *}*

Sends keys to the currently active window.

Example

- Win.Keys("{to *}%{f4}")
  - Sends alt-f4 to the current window, closing it.

{to activebar}

Sends the keys to the last window referenced by a button on an active windows bar.

Example

- Win.Keys("{to activebar}quick brown fox")
  - Sends "quick brown fox" to the last window selected via a click on an active windows bar.

{to autorun}

Sends the keys to the last window matched by an autorun-type command list.

Example

- Win.Keys("{to autorun}%{sp}n")
  - Sends alt-f4 to the current window, closing it.

{to captionlist}

Sends the keys to the first window matching the captionlist.

For full list of accepted window id types, see:

- *PPSR > Language Reference > Caption Lists*

- *PPSR > Language Reference > Window ID's*

Examples

- Win.Keys("{to *Firefox}^l")
  - Sends Ctrl-L to Firefox to go to the location bar.
- Win.Keys("{to c=metapad}quick brown fox")
  - Sends "quick brown fox" to the first window of class "metapad" found.
- Win.Keys("{to =firefox}"++myusername++"{ta}{w3}"++mypassword++"{en}")
  - Sends a username & password to Firefox.

{to =fpath}	Sends keys to program run from the file path, fpath. File path must be prepended with equals sign. ***Example*** - Win.Keys("{to =c:\pgms\pgm.exe}something to be sent") - Sends "something to be sent" to pgm.exe.
{to none}	Does not check if a window is available, instead makes keys available in keyboard buffer only. ***Example*** - Win.Keys("{to none}"++date) - Puts the current date on the kb buffer.

{to folder}fpath	Used with File Open/Save As dialogs.
	
	Changes the current directory of a File Open/Save As dialog to the specified folder path, fpath.
	
	Example
	
	- Win.Keys(?"{to folder}c:\program files")
	  - Changes the current directory of a file open/save dialog to: "c:\program files".
	  - Assign this command to a button on a bar of similar buttons with folder shortcuts, in order to create a quick-jump bar to be shown whenever a File Open/Save As dialog is shown.
	
	Notes
	
	- The process PowerPro takes to perform this action is:
	  - select the File Name field
	  - save anything currently in it
	  - send the folder path to the dialog
	  - restore the File Name field's contents
	- Generates a PowerPro error if used on anything other than a File Open or Save dialog.
	- Any text after fpath will be considered part of the folder path (and thus may cause errors).

{filemenu filepath}	Shows a menu created from the contents of a file, & the selected option is sent as keys to the active document. filepath=full path to the file containing the menu to

be shown

Example

- Win.Keys(?"{filemenu c:\path\to\menu.txt}")
  - Shows the contents of "menu.txt" as a menu, & sends the selection to the active document

Notes

- *Format commands can be used with menu text files, but require an @label at the start of the line (for some reason..?)

See Also

- More information on file menus:
  - *PowerPro Help CHM > Built-in Commands > *Menu > File menus*

{from c:\path\filename.txt}	Sends the contents of the specified text file to the active document. ***Example*** - Win.Keys(?"{from c:\mydocs\signature.txt}") ***Notes*** - The text file can contain both standard standard ascii characters & PowerPro key codes, enabling functionality such as text formatting, program shortcuts & menu actions. - Text file must be in DOS or Windows format (not Unicode). - Any text after the closing brace will cause errors.

Window Prefixes

- An optional single character prepended to the target portion of the target window string, with special meaning as outlined below:
- **Possible Values**

^	hides error messages
+	waits up to 3 secs for for the specified window to be ready
-	PowerPro sleeps a short time before sending the keys

- ***Examples***
  - Win.Keys("{to +*metapad}quick brown fox")
    - Waits up to 3 seconds for the first Metapad window found to be ready for input, then sends the keys "quick brown fox" to it.
  - Win.Keys("{to ^=firefox}^l{w1}miinx.com.au{en}")
    - Sends the F6 key to the first Firefox window found to select the location bar, hiding any error message if the window is not found, then waits 1/10th of a second before sending the url "miinx.com.au" & the Enter key.

keys_to_send	(string) any combination of alphanumeric characters & special keycodes making up the keys to be sent to the target window

Keycodes

- See:
  - *PPSR > Appendix > Win.Keys() & Win.SendKeys() Key Codes*

Notes

- Win.Keys() allows up to 1000 characters.
  - my testing doesn't show this... maybe 1000 bytes?

See Also

- *PPSR > Plugins > The Plugins > Win > Commands > by Category > Keys > Keys*
- *PPSR > Plugins > The Plugins > Win > Commands > by Category > Keys > SendKeys*

Macro

Overview

- Keyboard macros let you create predefined text shortcuts that, when typed, are replace by the full text.

Notes

- There are no functions related to the Macro command.

- The *Macro command can only be used with hot keys, and therefore does not appear in the PProConfig GUI command drop-down list. Using *Macro in any other context will generate an error message.

See Also

- For more information on creating & using PowerPro macros, see:
    - *PowerPro Help CHM > Built-in Commands > *Macro > *Macro command*

Menu

Overview

- The Menu command is used to display a menu.

Actions

Explorer	Shows a menu of folders recently accessed with Explorer.	
	Folder	Shows a menu of files from a folder.
Recent	Shows a menu of recently executed windows (and/or commands), & the selected item is re-executed.	
StartMenu	Shows the Windows start menu at the mouse cursor.	
	Show	Shows a command list as a menu, positioned according to specified keyword.
ShowAtButton	Meant for use on bar buttons, this command shows a menu aligned with the button used to display it.	
ShowAtCursor	Shows a command list as a menu at a text cursor, if present.	
ShowFile	Shows the contents of a file as a menu.	
	Tray	Shows a menu of all current tray icons.
UnderMouse	Show the menu of the window under the mouse. Windows 95/98 only.	

Message

Overview

- The old format *Message command is replaced by the built-in function messagebox() in the new expression syntax.

See Also

- For more information on the messagebox() function, see:
    - *PPSR > Built-in Functions > by Category > Dialogs > messagebox*

Prev Up Next

PPSR | *Built-in Commands* | *Mouse*

Mouse

Overview

- The old format *Mouse command becomes Win.Mouse() with PowerPro's new expression syntax.

Syntax

- **Win.Mouse(*mouse_codes*)**

Parameters

mouse_codes — (string) any combination of long-format or short-format codes, as a quote-delimited string

Possible Values

Modifier Keys

Alt		Reverses the Alt key (presses it if it's up, releases it if down).
	short form: *al*	
Ctrl		Reverses the Control key (presses it if it's up, releases it if down).
	short form: *ct*	
Shift		Reverses the Shift key (presses it if it's up, releases it if down).
	short form: *sh*	
Win		Reverses the Windows key (presses it if it's up, releases it if down).
	short form: *wi*	

Mouse Buttons

LeftClick		left click (both left down and left up)
	short form: *lc*	
LeftDown		left down
	short form: *ld*	
LeftUp		left up
	short form: *lu*	
LeftDouble		double left click

	short form: *ll* Note: "lc lc" will not work.
MiddleClick	middle click (both middle down and middle up) short form: *mc*
MiddleDown	middle down short form: *md*
MiddleUp	middle up short form: *mu*
MiddleDouble	double middle click short form: *mm* Note: "mc mc" will not work.
RightClick	right click (both right down and right up) short form: *rc*
RightDown	right down short form: *rd*
RightUp	right up short form: *ru*
RightDouble	double right click short form: *rr* Note: "rc rc" will not work.

- **Mouse Position**

Move x y	Moves the mouse x pixels right, & y pixels down. x & y can be negative. short form: *mo x y*
Save	Saves the current mouse position. short form: *sa*
Screen x y	Moves the mouse to a position (x,y) on the screen, where (0,0) is top left. short form: *ab x y*
Relative x y	Moves the mouse to a position x pixels to the right & y pixels below the top-left corner of the active window.

	short form: *re x y*
Restore	Restores a saved mouse position. If there is no saved position, the mouse is moved to screen (0,0). This mouse code has no short form.

Examples

- Win.Mouse("relative 50 100 leftclick")
  - Moves the mouse to the position (50,100) relative to the active window, then left-clicks.
  - This could also be written:
    Win.Mouse("re 50 100 lc")

- Win.Mouse("sa mo 100 40 lc restore")
  - Saves the current mouse position, then moves the mouse 100px right & 40px down before left-clicking, then restores the original mouse position.

- Win.Mouse("ct lc ct")
  - Sends a Ctrl-Left-click to the active window.

See Also

- *PPSR > Plugins > The Plugins > Win > Services - Alpha-order [M-Z] > Mouse*

PrevUp Next

Note

Overview

- The Note command lets you work with PowerPro notes.

Actions

CloseCategory	Closes all notes open from specified category.	
	DeleteOpenCategory	Deletes notes opened from specified category.
Open	Creates a new note or opens an existing note (or notes) & returns the note's handle.	
OpenCategory	Opens all notes in specified category.	
	OpenMenu	Shows a menu of all open notes, & brings the selected note to the top.
OpenOneFromMenu	Shows a menu of notes from specified category & opens the selected note.	
OpenToday	Shows notes with date categories today or before	
	ShowHideOpen	Toggles hidden/shown state of open notes.
ShowOpen	Shows all hidden notes from a category.	

See Also

- *PowerPro Help CHM > Built-in Commands > *Note > *Note command*
- *PPSR > Plugins > The Plugins > Note*

ScreenSaver

Overview

- The ScreenSaver command is used to work with the Windows screen saver.

Actions

Change	Changes to a random screen saver file (.scr file) in same folder.	
ChangeRestart	Sets, clears, or reverses the "Restart running saver if changed" setting in PowerPro Config > GUI Control dialog.	
ChangeTimeout	Changes saver timeout (time before saver kicks in) to the specied value.	
ChangeTo	Changes screen saver to file specified.	
	Disable	Disables the screen saver.
Enable	Enables the screen saver.	
	Start	Starts the screen saver.
Stop	Stops the screen saver.	
	TempDisable	Disables the saver until the mouse is moved.

Notes

- If the current screensaver is set to "none" or "blank" via the Contol Panel > Display Properties > Screen Saver dialog, PowerPro ScreenSaver commands will not work, & the screensaver "scrnsave.scr" in Windows' system32 directory is used.

Script

Overview

- The old format Script command encompassed a range of functions relating to control & manipulation of PowerPro scripts.
- Most of these commands are now deprecated for scripting use, and the rest are all mentioned in other places of this reference.

General Syntax for Script commands

- ***Script [command] [parameters]**
  - Script commands use the old Literal syntax.

Commands

assign		
	break	
close		
	closeforce	
debug		
	else	
elseif		
	endif	
endfor		
	flag	PP has 32 flags, numbered 0-31 Flags can only have a value of 0 or 1
for		
	global	

if (expr)	
	if (expr) do
jump	
	local
quit	
	quit all
quit (exp)	
	path
run	
	runfile
setstring	
	static

PrevUp Next

Shutdown

Overview

- The Shutdown command has functions to shutdown/exit both PowerPro and Windows.

Actions

Dialog	Shows the standard "Shut Down Windows" exit dialog.	
	Hibernate	Puts the computer into hibernate power mode.
LockWorkStation	Sign off user and lock workstation.	
	Logoff	Logs the current Windows user off.
PowerPro	Exits PowerPro.	
	Reboot	Shuts down Window and reboots the system.
Restart	Shuts down system with warm Windows restart.	
	Suspend	Puts the computer into suspend/standby power mode.
Windows	Shuts down windows.	

Timer

Overview

- Powerpro has 26 customisable timers, identified by the single-letter labels a, b, c, ..., z.

Actions

Autosave	Changes the interval between which timers are autosaved to the pcf file.	
Clear	Zeros (clears) the specified timer or timers.	
	Set	Starts, stops or toggles the specified timer(s) and sets the value.
Start	Starts the specified timer or timers.	
	StartStop	Toggles the running of the specified timer or timers.
Stop	Stops the specified timer or timers.	

See Also

- *PowerPro Help CHM > Built-in Commands > *Timer*
- *PowerPro Help CHM > Configuring with the GUI dialogs > Timers*
- For more precise timing, see the built-in functions perfcount & perffreq:
  - *PPSR > Built-in Functions > Functions > by Category > Timers > perfcount*
  - *PPSR > Built-in Functions > Functions > by Category > Timers > perffreq*

TrayIcon

Overview

- The *TrayIcon command is used to simulate mouse clicks on tray icons from any program.

Actions

Dump	Generates a file in the PowerPro folder called "trayicons.txt", containing a list of all tray icons.	
Hide	Hides specified tray icon from PowerPro & system tray.	
	Left	Simulates a left mouse click on the specified tray icon.
LeftDouble	Simulates a double left mouse click on the specified tray icon.	
Refresh	Refreshes the tray icons.	
	Right	Simulates a right mouse click on the specified tray icon.
RightDouble	Simulates a double right mouse click on the specified tray icon.	
Show	Forces a tray icon to be shown in both PowerPro's & the system's tray.	

See Also

- *PowerPro Help CHM > Built-In Commands > *TrayIcon > Working with tray icons from other programs*

Vdesk

Overview

- The Vdesk command works with PowerPro's virtual desktops feature.

Actions

Arrange	Displays a PowerPro Virtual Desktop Arrange window showing all desktops & windows on them.	
Clear	Clears the current desktop (closes all windows on it).	
	ClearAllClose	Move all windows to current desktop and then closes them.
Consolidate	Move all windows to current desktop & closes all other desktops.	
CreateOrSwitchTo	Switches to the specified desktop, if it exists, otherwise creates a new desktop named after the specified command list, then runs the commands on the list to populate the desktop.	
Lock	Locks all visible windows matching the specified caption list, so that they are visible on all desktops.	
Menu	Displays the virtual desktop menu.	
	MoveActive	Moves the active window to the specified desktop.
MoveAll	Moves all windows on the source desktop matching the specified caption list to the destination desktop.	
MoveAutorun	Moves last autorun window to the specified desktop.	
	New	Creates a new desktop.
NewFromList	Switches to the specified desktop, if it exists, otherwise creates a new desktop named after the specified command list, then runs the commands on the list to populate the desktop.	
Next	Activates the next virtual desktop.	
		Activates the previous

	Previous	virtual desktop.
ShowMenu	Shows a menu of desktops & windows, then either: ❍ the selected window is moved to the current desktop. ❍ the selected desktop is activated.	
SwitchMenu	Shows a menu of desktops and windows, & the selected one is activated.	
SwitchTo	Switches to the specified desktop.	
	ReplaceByList	Clears the current desktop, renames it to the specified command list, then runs the commands on the named list to populate the desktop.
Unlock	Unlocks all windows matching the specified caption list, so that they are only visible on one desktop.	

See Also

- *PowerPro Help CHM > Built-in Commands > *Vdesk*

PrevUp Next

Wait

Overview

- The Wait command is used to wait for some condition before continuing. The condition can be an amount of time, an expression, or both, depending on the command & action used.
- Waits should always be used after commands which act on other windows, as after PowerPro sends the window a message asking it to do something, the window itself must have time to respond.

Actions

Activity	Waits for mouse or keyboard activity.
	For Wait for specified length of time, or for an expression to evaluate to true.
ForInterval	Waits for specified amount of time.
	Message Displays a message box with specified msg and, optionally, a countdown timer starting at n seconds, & waits for user to press a button to determine whether to continue.
Quit	Quit all waits.
	Ready Wait for specified program(s) to be ready to accept input.
Until	Wait until specified length of time, or until an expression evaluates to true.

See Also

- PowerPro offers a few other types of waits:
  - the Event Plugin
    - *PPSR > Plugins > The Plugins > Event*
  - File.RunCallBack() in the File plugin
    - *PPSR > Plugins > The Plugins > File > Services > RunCallBack*

- File.RunWait() in the File plugin
  - *PPSR > Plugins > The Plugins > File > Services > RunWait*

Prev Up Next

Wallpaper

Overview

- The Wallpaper command is used to change the desktop wallpaper.

Actions

Change	Changes the wallpaper to another file from the same folder as the current desktop wallpaper file.
ChangeTo	Changes the desktop wallpaper to the specified file, & saves the details in the current .pcf.
Show	Changes the desktop wallpaper to the specified file, but does not save the new file details in the current .pcf.
Style	Sets the wallpaper file layout

Notes

- .bmp, .jpg & .jpeg files may be used as desktop wallpaper.

PrevUp Next

PPSR | *Built-in Commands* | *Window*

Window

Overview

- The Window commands allow manipulation of the windows of running programs.

Actions

AutoMin	Minimises all windows matching the specified caption list in one of 2 ways depending on whether each individual window is listed in the "Auto tray min" list on the PowerPro Config GUI > Setup tab or not, either minimising the window to the tray, or performing an ordinary minimise.	
Back	Sends all windows matching the specified caption list to the bottom of the stack of displayed windows.	
BackShow	Toggles backmost/foremost setting of each window matching the specified caption list, sending it to the back if foremost, else activating it.	
Center	Centres the specified window(s) within the full screen area.	
Close	Closes the specified window(s).	
	Close2	Closes the specified window(s) using a different technique than Window.Close().
CloseForce	Forces the specified window(s) to close, possibly losing any unsaved information.	
Hide	Makes a window invisible.	
	HideShow	Toggles a window's visibility.
Max	Maximises the specified window(s).	
	MaxNormal	Toggles the maximised state of the specified window(s).
Min	Minimises the specified window(s).	

	MinMemory	Sets the memory working set for the specified window(s). Windows NT only.
MinRestore	Toggles the minimised state of each window matching the specified caption list.	
Move	Sets the specified window(s) to move with the mouse until any mouse button is pressed.	
Normal	Restores all windows matching the specified caption list to "Normal" size (not minimised or maximised).	
NotTop	Removes any "Always On Top" settings for all windows matching the specified caption list.	
OnTop	Sets all windows matching the specified caption list to be "Always on top".	
PostMessage	Sends a WinAPI PostMessage() call to each window matching the specified caption list.	
Position		
	Rollup	Toggles a window's "rolled-up" state, rolling it up to just its caption if it is fully visible, or showing it again if it is already rolled-up.
SendMessage	Sends a WinAPI SendMessage() call to each window matching the specified caption list.	
SetPriority	Sets process priority of specified window.	
	Show	Activates the specified window & shows it if hidden.
Size	Enables the user to size the window by moving the mouse; click any mouse button to stop.	
TopNotTop	Toggles the "Always on top" setting for each window matching the specified caption list.	
Trans	Makes all windows matching the specified caption list transparent.	

	Windows 2000 and XP only.
TransMouse	Makes all windows matching the specified caption list transparent, as well as making all mouse clicks pass through them. Windows 2000 and XP only.
TrayMin	Minimizes all windows matching a caption list to the system tray.

Specifying the Window Id

- The window identifier (wid parameter) can be:
  - Any standard caption list item
    - See:
      - *PPSR > Language Reference > Caption Lists*
      - *PPSR > Language Reference > Window Id's*
  - Extra wid keywords for Window.[Action]() commands

*	The active window.
all	All windows.
hidden	Hidden windows.
menu	Displays a menu of active windows allowing user to select one to receive the command. Put "noembed" (must be lowercase) after keyword to..?
menu hidden	Similar to "menu", except the list includes active, hidden, & tray-minimised windows.
menu onlyhidden	Similar to "menu", except the list only includes hidden windows.
menu trayminned	Similar to "menu", except the list includes active & tray-minimised windows. (No hidden windows, though.)
menux	Similar to menu, except that if only one window matches the specified captionlist, then command is executed on that window without showing the menu.

Avoiding Errors

- Put an underscore _ after the action name to avoid an error message if the window does not exist.

- E.g.

    - Window.Close_("*notepad*)

        - Closes any open notepad windows, but does nothing if there are none open.

- ***Notes***

    - Underscore only works with the Window built-in command.
    - Underscore only works with expression syntax -- for Literal Syntax, use an exclamation mark (!) instead.

Prev Up Next

Bar.Close

Description

- Closes a bar and removes it from memory.

Syntax

- **Bar.Close(*barname*)**

Parameters

barname	(string) name of the bar

Examples

- Bar.Close("activeBar")
  - Closes the bar named "activeBar".

Bar.Format

Description

- Formats a bar.

Syntax

- **Bar.Format(*barname, sKeywords*)**

Parameters

barname	(string) name of the bar to format
sKeywords	(string) any combination of the format keywords as a space-separated list

Possible Values

autohide n	where n=number of milliseconds to hide after Note that "Show if bump" in the PPConfig Command List Properties window must be set to something other than "None", otherwise the bar will not appear on screen edge bump.
back bgImgPath	where bgImgPath=path to the background image, using forward slashes as the path delimiter, & in quotes. Since the parameter sKeywords is itself in quotes, the inner pair must be escaped. Changes the background to file.bmp or use back none to remove background; put file name in double quotes if it contains blanks. Use with "back2" to toggle between two backgrounds each time the command is executed. Use "back none" to specify no background image. Always add the "refresh" keyword when changing a bar's background, in order to show the change.
back2 bgImgPath	See "back".
position n	Position n is the nth item in the menu of positions that appears in PPCD drop-down for Bar Position. There are 23 possible values for n:

1	floating
2	locked
3	left caption
4	middle caption
5	right caption
6	left screenbar
7	top screenbar
8	right screenbar
9	bottom screenbar
10	taskbar

11	taskbar no start
12	fix top right
13	fixed top right offset
14	fixed top center
15	fixed top left
16	fixed left center
17	fixed bottom left
18	fixed bottom center
19	fixed bottom right
20	left of active
21	right of active
22	above active
23	below active

refresh	Closes & reopens the bar. Use with "back", "back2" & "position" keywords to see the changes.

Examples

- Bar.Format("DocsBar", "autohide 500")
  - Sets the bar named "DocsBar" to automatically hide after 500ms when the mouse is moved off it.
- Bar.Format("DocsBar", "autohide -500")
  - Toggles the bar named "DocsBar" between automatically hiding after 500ms when the mouse is moved off it, and not automatically hiding.
- Bar.Format("Links", "back \"C:/graphics/lines.bmp\" refresh")
  - Each time this command is called, it will toggle the background of bar "Links" between "c:\graphics\lines.bmp" and no background.
- Bar.Format("Links", "back \"C:/graphics/lines.bmp\" back2 none refresh")
  - Each time this command is called, it will toggle the background of bar "Links" between "c:\graphics\lines.bmp" and no background.
- Bar.Format("Places", "position 22 refresh")
  - Sets the bar named "Places" to be above the active window.
- Bar.Format("Places", "position -23 refresh")
  - Toggles the bar named "Places" between its current position & being positioned "Below active", refreshing it to show the change.

Notes

- The resulting new bar configuration is always saved in the .pcf file.

PrevUp Next

Bar.Hide

Description

- Hides a bar but keeps it in memory (for faster reshow).

Syntax

- **Bar.Hide(*barname*)**

Parameters

barname	(string) name of the bar to format

Examples

- Bar.Hide("Tools")
  - Hides the bar named "Tools".

Prev Up Next

Bar.HideShow

Description

- Hides a bar if visible; shows it otherwise.

Syntax

- **Bar.HideShow(*barname*)**

Parameters

barname	(string) name of the bar to format

Examples

- Bar.HideShow("Tools")
  - Toggles the visibility of the bar named "Tools", showing it if it's hidden & hiding it otherwise.

Bar.Keys

Description

- Readies bar to receive keys.

Syntax

- **Bar.Keys(*barname*)**

Parameters

barname	(string) name of the target PowerPro bar

Keystrokes

left arrow	move to next button
right arrow	move to previous button
up arrow	move to next row in multi-row bar
down arrow	move to previous row in multi-row bar
esc	return mouse cursor to position preceding the Bar.Keys() command
enter	left-click current button
ctrl+enter	show config dialog
L	left-click current button
M	middle-click current button
R	right-click current button
tab	move to next button
home	move to first button
end	move to last button

Examples

- Bar.Keys("myBar")
  - Moves the mouse to the start of the bar named "myBar".

Prev Up Next

Bar.SelectSubBar

Description

- Shows the specified subbar on a PowerPro bar, hiding any other subbars.

Syntax

- **Bar.SelectSubBar(*"barname @subbarname"*)**

Parameters

barname	(string) name of the bar containing the subbar to show
subbarname	(string) name of the subbar to show

Examples

- Bar.SelectSubBar("PPBar @sub1")
  - Shows subbar "sub1" on bar "PPBar".

See Also

- A "subbar" is a portion of a PowerPro bar. For more info on subbars, see:
  - *PowerPro Help CHM > Miscellaneous Usage Topics > "Displaying different subsets of a bar" for more information.*
- To get the name of the currently selected subbar, use the built-in PowerPro function subbarname. See:
  - *PPSR > Built-in Functions > by Category > PowerPro > subbarname*

Bar. SelectSubBarToButton

Description

- Shows the specified subbar aligned to the bar button pressed most recently.

Syntax

- **Bar.SelectSubBarToButton(*"barname @subbarname"*)**

Parameters

barname	(string) name of the bar containing the subbar to show
subbarname	(string) name of the subbar to show

Examples

- Bar.SelectSubBarToButton("Test @subbar1")
  - Shows the subbar "subbar1" from bar "Test" aligned to the button just pressed.

Notes

- The subbar is shown aligned with the bottom of the pressed button.
- If the button pressed is on the same bar as subbarname, the bar is redraws itself at the location of the pressed button.
- If there are commands not within a subbar on the bar specified by barname, they are also shown when the subbar part of the bar is shown.

See Also

- To get the name of the currently selected subbar, use the built-in PowerPro function subbarname. See:
  - *PPSR > Built-in Functions > by Category > PowerPro > subbarname*

Prev Up Next

Bar. SelectSubBarToMouse

Description

- Shows the specified subbar aligned to the mouse.

Syntax

- **Bar.SelectSubBarToMouse(*"barname @subbarname"*)**

Parameters

barname	(string) name of the bar containing the subbar to show
subbarname	(string) name of the subbar to show

Examples

- Bar.SelectSubBarToMouse("Test @subbar1")
  - Shows the subbar "subbar1" from bar "Test" aligned to the mouse.

See Also

- The Notes & See Also sections for SelectSubBarToButton above.

Bar.Show

Description

- Shows a bar, optionally using slide animation to do so.

Syntax

- **Bar.Show(*barname*)**
  - shows bar "barname"

- **Bar.Show(*"*keyword", barname*)**
  - shows bar "barname" with settings given by "*keyword"

Parameters

***keyword**	(string) slide direction and/or mouse position

Possible Values

Slide animations

*"*vertical"*	Powerpro determines the slide direction depending on which half of the screen the mouse is positioned in, top or bottom.
*"*horizontal"*	Powerpro determines the slide direction depending on which half of the screen the mouse is positioned in, left or right.
*"*fromtop"*	Bar slides in from top.
*"*frombottom"*	Bar slides in from bottom.
*"*fromleft"*	Bar slides in from left.
*"*fromright"*	Bar slides in from right.
*"*none"*	Removes any preset slide animation.

Mouse-related

*"*move"*	The mouse cursor is positioned over the bar. "*move" can be combined with a

	slide direction keyword in the same Bar.Show() call, in either order.
barname	(string) name of the bar to show

Examples

- Bar.Show("*vertical", "ActiveWindows")
  - Shows the bar named "ActiveWindows" sliding down from the top if the mouse is in the top half of the screen, and sliding up from the bottom if the mouse is in the bottom half of the screen.
- Bar.Show("*move *vertical", "ActiveWindows")
  - Shows the bar named "ActiveWindows" sliding down from the top if the mouse is in the top half of the screen, and sliding up from the bottom if the mouse is in the bottom half of the screen, and moves the mouse to the start of the bar.

Notes

- Only works with Windows 98, 2000 and later.
- The Command Lists > Setup checkbox "Use slide animation" does not have to be checked for the Bar.Show() command to work.

PrevUp Next

Bar.ToMouse

Description

- Temporarily moves the specified bar to the mouse, showing it if hidden.

Syntax

- **Bar.ToMouse(*barname*)**

Parameters

barname	(string) name of the bar to move

Examples

- Bar.ToMouse("Test")
  - Moves the bar named "Test" to the mouse.

Notes

- Only works with bars that have position: Floating.
- Usually used with a PowerPro hotkey.

Clip.Capture

Description

- Sets PowerPro's clipboard history tracking state.

Syntax

- **Clip.Capture(*keyword[, pathOrRefresh]*)**

Parameters

keyword

(string) action to perform

Possible Values

on	Turns clipboard history tracking on.
off	Turns clipboard history tracking off.
reverse	Toggles the state of clipboard history tracking.

pathOrRefresh

(string) specifies either a folder path or the keyword "refresh"

Optional.

Possible Values

a folder path	Changes PowerPro's clip capture storage folder to the specified folder.
refresh	Resets PowerPro's internal memory copy of clip files and dates. Used if a new text file is inserted into the clip folder without using clip capture.

Examples

- Clip.Capture("on")
  Win.Debug(cliptrackon)
  - Turns clip tracking on, then debugs the state of clip history tracking.
  - The debug window would show 0.

- Clip.Capture("reverse")
  - Toggles the state of PowerPro's clip history tracking, turning it on if it is off, & off if it is on.

- Clip.Capture("on c:/web/cliptest")
  - Sets the clip tracking folder to the specified path, which is defined with forward slashes as folder separators.

- Clip.Capture(?"on c:\web\cliptest")
  - Sets the clip tracking folder to the specified path, which is defined with backslashes as folder separators using ?c..c syntax.

- Clip.Capture(?"on c:\program files\powerpro\clip")
  - Sets the clip tracking folder to PowerPro's default clip folder.

- Clip.Capture("on "++pprofolder++"clip")
  - Also sets the clip tracking folder to PowerPro's default clip folder.

- Clip.Capture("on refresh")
  - Resets PowerPro's internal clips memory.

- Clip.Capture("off")
  File.AllFiles(path,"Clip.FilePaste(\"|\\n\\n\")")
  Clip.Capture("on")
  - Turns clip tracking off, performs a clip intensive operation, then turns clip tracking back on again, eliminating the overhead from PowerPro updating its clip folder & memory for each file during the process.

Notes

- PowerPro clip history tracking is initially on, by default.

PrevUp Next

Clip.ClearClip

Description

- Clears the clip board.

Syntax

- **Clip.ClearClip**

Examples

- Clip.ClearClip

Notes

- This is not working?

PrevUp Next

Clip.ClearRecent

Description

- Clear PowerPro's list of recently captured clips.

Syntax

- **Clip.ClearRecent**

Examples

- Clip.ClearRecent

PrevUp Next

Clip.Copy

Description

- Copies the selected text by sending Ctrl-C to the foreground window.

Syntax

- **Clip.Copy**

Examples

- Clip.Copy

Notes

- Clip.Copy doesn't actually return the selected text to the calling PowerPro script -- it just places the text on the clipboard.

Prev **Up** **Next**

Clip.CopyThenToFile

Description

- Send Ctrl-C to copy the selected text, then copies the clipboard contents to a file.

Syntax

- **Clip.CopyThenToFile(*filepath*)**

Parameters

filepath	(string) path to the target file Must be a full filepath including file name & extension.

Examples

- Clip.CopyThenToFile("c:/web/out.txt")
  - Copies the selected text to the clipboard, then to the file "out.txt".

Notes

- PowerPro will only copy plain text across to the file, removing any rich text or other formatting from the selection.

Prev Up Next

Clip.Cut

Description

- Cuts the selected text by sending Ctrl-X to the foreground window.

Syntax

- **Clip.Cut**

Examples

- Clip.Cut

Notes

- Clip.Cut doesn't actually return the cut text to the calling PowerPro script -- it just cuts the text (if it can) from the foreground window & places it on the clipboard.

Prev Up Next

Clip.Delete

Description

- Shows the clipboard history list, then deletes the entry selected by the user.

Syntax

- **Clip.Delete**

Examples

- Clip.Delete

Clip.File

Description

- Copies a file to the clipboard.

Syntax

- **Clip.File(*filepath*)**

Parameters

filepath	(string) path to the target file

Examples

- Clip.File(?"c:\web\out.txt")
  - Puts the contents of the file "out.txt" onto the clipboard.

Clip.FilePaste

Description

- Copies a file to the clipboard, then pastes it by sending Ctrl-V to the foreground window.

Syntax

- **Clip.FilePaste(*filepath*)**

Parameters

filepath	(string) path to the target file

Examples

- Clip.FilePaste(?"c:\web\out.txt")
  - Puts the contents of the file "out.txt" onto the clipboard, then pastes it to the active window.

Prev Up Next

Clip.FileIPaste

Description

- Copies a file to the clipboard, then pastes it by sending Ctrl-Ins to the foreground window.

Syntax

- **Clip.FileIPaste(*filepath*)**

Parameters

filepath	(string) path to the target file

Examples

- Clip.FileIPaste(?"c:\web\out.txt")
  - Puts the contents of the file "out.txt" onto the clipboard, then pastes it to the active window.

Notes

- This command has an "i" in it.

Prev Up Next

Clip.LongDate

Description

- Puts the date onto the clipboard in the system's long date format.

Syntax

- **Clip.LongDate**

Examples

- Clip.LongDate

Clip.Menu

Description

- Shows a menu of recently captured clips (or, optionally, a menu of clips from another folder), allowing user to select an entry to put on the clipboard.

Syntax

- **Clip.Menu(*foldername*)**

Parameters

foldername	(string) name of the folder to show clips from Optional, default is the current PowerPro clip capture folder (usually the "clip" folder under the PowerPro installation directory).

Examples

- Clip.Menu
  - Shows a menu of recent clips & puts selection onto clipboard.

- Clip.Menu("work")
  - Shows a menu of the clips from the folder "work" (a subfolder of the current clip capture folder) & pastes the selection to the active window.

Note

- Clip.Menu does not paste the selected entry -- use Clip.MenuPaste or Clip.MenuIPaste for that.

See Also

- For more information on setting up subfolders for permanent clip storage, see:
  - *PowerPro Help CHM > Configuring with the GUI Dialogs > Clip Filters*

PrevUp Next

Clip.MenuPaste

Description

- Shows a menu of recently captured clips (or, optionally, a menu of clips from another folder), & the selected entry is put onto the clipboard then pasted into the active window using Ctrl-V.

Syntax

- **Clip.MenuPaste(*foldername*)**

Parameters

foldername	(string) Name of the folder to show clips from, which must be a subfolder of the current clip capture folder. Optional, default is the current PowerPro clip capture folder (usually the "clip" folder under the PowerPro installation directory).

Examples

- Clip.MenuPaste
  - Shows a menu of recent clips & pastes selection to active window.

- Clip.MenuPaste("work")
  - Shows a menu of the clips from the clip subfolder "work" & pastes the selection to the active window.

See Also

- For more information on setting up subfolders for permanent clip storage, see:
  - *PowerPro Help CHM > Configuring with the GUI Dialogs > Clip Filters*

Clip.MenuIPaste

Description

- Shows a menu of recently captured clips (or, optionally, a menu of clips from another folder), & the selected entry is put onto the clipboard then pasted into the active window using Ctrl-Ins.

Syntax

- **Clip.MenuIPaste(*foldername*)**

Parameters

foldername	(string) Name of the folder to show clips from, which must be a subfolder of the current clip capture folder. Optional, default is the current PowerPro clip capture folder (usually the "clip" folder under the PowerPro installation directory).

Examples

- Clip.MenuIPaste
  - Shows a menu of recent clips & inserts selection in active document.

- Clip.MenuIPaste("work")
  - Shows a menu of the clips from the clip subfolder "work" & inserts the selection to the active window.

Notes

- Note that this keyword contains an added letter "i".

See Also

- For more information on setting up subfolders for permanent clip storage, see:
  - *PowerPro Help CHM > Configuring with the GUI Dialogs > Clip Filters*

PrevUp Next

Clip.Reattach

Description

- Puts PowerPro at the front of the queue of applications tracking clipboard history.

Syntax

- **Clip.Reattach**

Examples

- Clip.Reattach

Clip.Paste

Description

- Sends Ctrl-V to the foreground window to paste the clipboard's contents.

Syntax

- **Clip.Paste**

Examples

- Clip.Paste

Prev **Up** **Next**

Clip.ShortDate

Description

- Puts the date onto the clipboard in the system's short date format.

Syntax

- **Clip.ShortDate**

Examples

- Clip.ShortDate

Clip.Text

Description

- Puts specified text onto the clipboard.

Syntax

- **Clip.Text(*sText*)**

Parameters

sText	(string) a single line of text

Examples

- Clip.Text(exefullpath)
  - Puts the full path to exe for the current foreground window onto the clipboard.

- Clip.Text("quick brown fox")
  - Puts the string "quick brown fox" onto the clipboard.

Notes

- This command can only work with a single line of text. Use the Clip plugin to work with multiple lines of text.

Prev Up Next

Clip.TextAppend

Description

- Appends specified text to the clipboard.

Syntax

- **Clip.TextAppend(*sText*)**

Parameters

sText	(string) a single line of text If sText is empty (""), a new line is appended to the clipboard.

Examples

- Clip.TextAppend(currentdir)
  - Appends the current working directory path of the active window to the clipboard.

- Clip.TextAppend("")
  - Appends a new line character to the clipboard.

Notes

- This command can only work with a single line of text. Use the Clip plugin to work with multiple lines of text.

Prev Up Next

Clip.TextPaste

Description

- Put specified text onto the clipboard, and then pastes it to the active window with Ctrl-V.

Syntax

- **Clip.TextPaste(*sText*)**

Parameters

sText	(string) a single line of text

Examples

- Clip.TextPaste(pprofolder)
  - Puts PowerPro's folder path to the clipboard then pastes it to the active window.

Notes

- This command can only work with a single line of text. Use the Clip plugin to work with multiple lines of text.
- For long text strings, a Clip.TextPaste() command will often be faster than a Win.Keys() command to paste into the current document.

Clip.Time

Description

- Puts the current time onto the clipboard.

Syntax

- **Clip.Time**

Examples

- Clip.Time

PrevUp Next

Clip.ToFile

Description

- Copies the clipboard's contents to the specified text file.

Syntax

- **Clip.ToFile(*fpath*)**

Parameters

fpath	(string) path to the target file

Examples

- Clip.ToFile(?"c:\docs\stuff.txt")
  - Copies the clipboard to "stuff.txt".

Notes

- The entire contents of the text file are replaced by the clipboard contents.

Clip.ToFileAppend

Description

- Appends the clipboard's contents to the specified text file.

Syntax

- **Clip.ToFileAppend(*fpath*)**

Parameters

fpath	(string) path to the target file

Examples

- Clip.ToFileAppend(?"c:\docs\stuff.txt")
  - Appends the clipboard to the contents of the file "stuff.txt".

Clip.Write

Description

- Sets the state of writing of clips to memory.

Syntax

- **Clip.Write(*keyword*)**

Parameters

keyword	(string) new setting for Clip writing functionality

Possible Values

"off"	turns off clip writing
"on"	turns on clip writing
"reverse"	toggles clip writing state

Examples

- Clip.Write("off")
  - Turns off standard capturing of clips by PowerPro.

Notes

- This command would be used if another clip management system was implemented by using the special ClipCaptured command list.

See Also

- *PPSR > Plugins > The Plugins > Clip*

Configure. CommandLists

Description

- Starts the PowerPro Config GUI at the CommandLists tab.

Syntax

- **Configure.CommandLists**

Examples

- Configure.CommandLists

Configure.GUI

Description

- Starts the PowerPro Config GUI at the GUI tab.

Syntax

- **Configure.GUI**

Examples

- Configure.GUI

Prev Up Next

Configure.ImportCL

Description

- Imports specified command list configuration text file into the current PowerPro configuration.

Syntax

- **Configure.ImportCL(*inifile*)**

Parameters

inifile	(string) name of the .ini file, optionally with path & .ini extension If inifile does not include the full path, then the file is assumed to be in the "config" subfolder of the folder containing the current .pcf file. Optional; if omitted, a PowerPro file browse dialog is shown.

Examples

- Configure.ImportCL
  - Shows a file browse dialog for selection of the config file to import.

- Configure.ImportCL(pprofolder++?"backups\lists.ini")
  - Imports the file lists.ini, which is located in "backups" under PowerPro's installation folder.

- Configure.ImportCL("list")
  - Imports the command list config file "lists.ini", which is located in the "config" subfolder of PowerPro's installation folder.

See Also

- *PowerPro Help CHM > Configuring with text files > Maintaining configuration using .ini files > Importing text configurations with commands*

PrevUp Next

Configure.ImportHot

Description

- Imports specified hotkeys configuration text file into the current PowerPro configuration.

Syntax

- **Configure.ImportHot(*inifile*)**

Parameters

inifile	(string) name of the .ini file, optionally with path & .ini extension If inifile does not include the full path, then the file is assumed to be in the "config" subfolder of the folder containing the current .pcf file. Optional; if omitted, a PowerPro file browse dialog is shown.

Examples

- Configure.ImportHot
  - Shows a file browse dialog for selection of the hotkeys config file to import.

- Configure.ImportHot(pprofolder++?"backups\hotkeys.ini")
  - Imports the file hotkeys.ini, which is located in "backups" under PowerPro's installation folder.

- Configure.ImportHot("hotkeys")
  - Imports the hotkeys config file "hotkeys.ini", which is located in the "config" subfolder of PowerPro's installation folder.

See Also

- *PowerPro Help CHM > Configuring with text files > Maintaining configuration using .ini files > Importing text configurations with commands*

PrevUp Next

Configure.ImportSched

Description

- Imports specified scheduled events configuration text file into the current PowerPro configuration.

Syntax

- **Configure.ImportSched(*inifile*)**

Parameters

inifile	(string) name of the .ini file, optionally with path & .ini extension If inifile does not include the full path, then the file is assumed to be in the "config" subfolder of the folder containing the current .pcf file. Optional; if omitted, a PowerPro file browse dialog is shown.

Examples

- Configure.ImportSched
  - Shows a file browse dialog for selection of the scheduled events config file to import.

- Configure.ImportSched(pprofolder++?"backups\events.ini")
  - Imports the file events.ini, which is located in "backups" under PowerPro's installation folder.

- Configure.ImportSched("events")
  - Imports the scheduled events config file "events.ini", which is located in the "config" subfolder of PowerPro's installation folder.

See Also

- *PowerPro Help CHM > Configuring with text files > Maintaining configuration using .ini files > Importing text configurations with commands*

PrevUp Next

Configure.Media

Description

- Starts the PowerPro Config GUI at the Media tab.

Syntax

- **Configure.Media**

Examples

- Configure.Media

Prev Up Next

Configure. NewReminderMessage

Description

- Starts the PowerPro Config GUI & opens an "Edit scheduled command" dialog preset to create a scheduled message.

Syntax

- **Configure.NewReminderMessage**

Examples

- Configure.NewReminderMessage

Prev Up Next

Configure.Scheduler

Description

- Starts the PowerPro Config GUI at the Scheduler tab.

Syntax

- **Configure.Scheduler**

Examples

- Configure.Scheduler

Configure.Setup

Description

- Starts the PowerPro Config GUI at the Setup tab.

Syntax

- **Configure.Setup**

Examples

- Configure.Setup

Configure.Timers

Description

- Starts the PowerPro Config GUI at the Timers tab.

Syntax

- **Configure.Timers**

Examples

- Configure.Timers

PrevUp Next

Configure. WriteAllToPCF

Description

- Write the entire current PowerPro configuration, including any imported text files, to a .pcf file.

Syntax

- **Configure.WriteAllToPCF(*pcfpath*)**

Parameters

pcfpath	(string) name of the target .pcf file, including path Optional; if omitted, a PowerPro file Save As prompt will be shown.

Examples

- Configure.WriteAllToPCF
  - Shows a PowerPro file prompt for the .pcf file to save to, then writes to it the current configuration.

- Configure.WriteAllToPCF(pprofolder++?"backups\"++date++".pcf")
  - Writes the current configuration to a .pcf file in the "backups" directory of PowerPro's installation directory, named with the current date.

Notes

- This command is intended to help debug text imports by showing what has been imported to memory.

- The saved configuration file can be viewed using the pproconf program.

Desktop.HideIcons

Description

- Hides all icons on the desktop.

Syntax

- **Desktop.HideIcons**

Examples

- Desktop.HideIcons

PrevUp Next

Desktop.HideTaskBar

Description

- Hides the Windows TaskBar.

Syntax

- **Desktop.HideTaskBar**

Examples

- Desktop.HideTaskBar

Prev Up Next

Desktop. HideShowIcons

Description

- Toggles the visibility of the desktop icons.

Syntax

- **Desktop.HideShowIcons**

Examples

- Desktop.HideShowIcons

PrevUp Next

Desktop. HideShowTaskBar

Description

- Toggles the visibility of the Windows TaskBar.

Syntax

- **Desktop.HideShowTaskBar**

Examples

- Desktop.HideShowTaskBar

Prev Up Next

Desktop. HideShowWindows

Description

- Toggles display of all windows on the desktop.

Syntax

- **Desktop.HideShowWindows**

Examples

- Desktop.HideShowWindows

Notes

- Desktop window visibility is toggled for all windows together as a group, not individually. (So, all are shown, or all or hidden.)

Prev Up Next

Desktop.IconTextColor

Description

- Set color of text under desktop icons; use the TransIconText auto function to reset color when background changes

Syntax

- **Desktop.IconTextColor(*r*,*g*,*b*)**
- **Desktop.IconTextColor(*"r g b"*)**
  - alternative syntax

Parameters

r, g & b	either: • (integer) 3 comma-separated integer values: *r* — red value — 0<=r<=255 *g* — green value — 0<=g<=255 *b* — blue value — 0<=b<=255 • (string) a space-separated string of 3 integers representing an rgb color value

Examples

- Desktop.IconTextColor(0,0,0)
  - Changes desktop icon text color to black.
- Desktop.IconTextColor("0 0 0")
  - Also changes desktop icon text color to black.
- Desktop.IconTextColor(255, 255, 255)

- Changes desktop icon text color to white.

PrevUp Next

Desktop. MinShowWindows

Description

- Toggles the minimised state of all windows on the desktop.

Syntax

- **Desktop.MinShowWindows**

Examples

- Desktop.MinShowWindows

Notes

- Desktop windows are toggled as a group, not individually.

Prev Up Next

Desktop.RestoreIcons

Description

- Restores the relative positions of desktop icons

Syntax

- **Desktop.RestoreIcons(*filename*)**

Parameters

filename	(string) name of the icon positions file, including the ".iconpos" extension

Examples

- Desktop.RestoreIcons("deskicons.iconpos")
  - Restores icon positions from the file "c:\program files\powerpro\deskicons.iconpos".

Notes

- All .iconpos files are kept in the Powerpro folder.

See Also

- *PowerPro Help CHM > Built-in Commands > *Desktop > Saving and restoring desktop icon positions*

Desktop.Resolution

Description

- Changes screen display resolution.

Syntax

- **Desktop.Resolution(*w, h, depth, freq, nosave*)**
  - Changes display resolution to w*h with specified depth & frequency.

- **Desktop.Resolution (*w1, h1, w2, h2, nosave*)**
  - Toggles display resolution between w1*h1 and w2*h2.

Parameters

w	(integer) new horizontal pixels
h	(integer) new vertical pixels
depth	(integer) new color depth Optional. **Possible Values** *4* *8* *16* *24*
freq	(integer) new refresh frequency Optional. Works on Windows NT only.

w1	(integer) 1st horizontal pixels setting, for toggling between two resolutions
h1	(integer) 1st vertical pixels setting, for toggling between two resolutions
w2	(integer) 2nd horizontal pixels setting, for toggling between two resolutions
h2	(integer) 2nd vertical pixels setting, for toggling between two resolutions
nosave	(string) stops new desktop settings being saved to the Windows Registry Optional, default is that settings are saved. **Possible Values** *"nosave"*

Examples

- Desktop.Resolution(1024, 768, 4, 300, "nosave")
  - Changes display resolution to 1024x768

- Desktop.Resolution (1024, 768, 800, 600)
  - Toggles display resolution between 1024x768 and 800x600.

See Also

- *PowerPro Help CHM > Built-in Commands > *Desktop > Changing screen display resolution*

Desktop.SaveIcons

Description

- Saves the relative positions of the desktop icons.

Syntax

- **Desktop.SaveIcons(*filename*)**

Parameters

filename	(string) name of the icon positions file, including the ".iconpos" extension

Examples

- Desktop.SaveIcons("deskicons.iconpos")

Notes

- All .iconpos files are kept in the Powerpro folder.

See Also

- *PowerPro Help CHM > Built-in Commands > *Desktop > Saving and restoring desktop icon positions*

Desktop.SaveIconsGrid

Description

- Aligns icons according to a grid & optionally also saves their relative position details to a file.

Syntax

- **Desktop.SaveIconsGrid (*xSpacing, ySpacing[, filename]*)**

Parameters

xSpacing	(integer) horizontal grid spacing
ySpacing	(integer) vertical grid spacing
filename	(string) name of the icon positions file, including the ".iconpos" extension Optional, default is to not save position details to a file but to just align the icons.

Examples

- Desktop.SaveIconsGrid(30, 20)
  - Aligns the desktop icons according to a grid with a relative horizontal spacing factor of 30 & a relative vertical spacing factor of 20.

- Desktop.SaveIconsGrid(30, 20, "deskicons.iconpos")
  - Aligns the desktop icons according to a grid with a relative horizontal spacing factor of 30 & a relative vertical spacing factor of 20, then saves the icons' relative positions to the file "deskicons.iconpos".

Notes

- All .iconpos files are kept in the Powerpro folder.

- Positions are stored as numbers that are independent of screen resolution. This enables relative icon positions to be maintained under different screen resolutions.

See Also

- *PowerPro Help CHM > Built-in Commands > *Desktop > Saving and restoring desktop icon positions*

PrevUp Next

Desktop.SetWorkArea

Description

- Sets the size of the maximum work area on the desktop screen.

Syntax

- **Desktop.SetWorkArea(*left, top, bottom, right*)**
- **Desktop.SetWorkArea(*"left top bottom right"*)**
  - Alternative syntax.

Parameters

left	(integer) left screen coordinate
top	(integer) top screen coordinate
bottom	(integer) bottom screen coordinate
right	(integer) right screen coordinate

Examples

- Desktop.setWorkArea("0 0 1024 747")
  - Sets work area to the rectangle bounded by top left coord (0,0) and bottom right coord (1024, 747).

Notes

- The desktop work area is equivalent to the space taken up by a maximized window.

See Also

- *PPSR > Plugins > The Plugins > Win > System > SetWorkArea*

PrevUp Next

Desktop.ShowIcons

Description

- Shows the desktop icons.

Syntax

- **Desktop.ShowIcons**

Examples

- Desktop.ShowIcons

Prev Up Next

Desktop.ShowTaskBar

Description

- Shows the Windows TaskBar.

Syntax

- **Desktop.ShowTaskBar**

Examples

- Desktop.ShowTaskBar

Prev Up Next

Desktop. ShowTaskBarAutoHide

Description

- Shows the TaskBar, then re-hides it when the mouse is moved away & the TaskBar is not the foreground window.

Syntax

- **Desktop.ShowTaskBarAutoHide**

Examples

- Desktop.ShowTaskBarAutoHide

Notes

- Assign this command to a hotkey that performs a screen bump in order to use an auto-hiding TaskBar but prevent inadvertent shows by movements near the screen bottom.

  Some systems may also require Desktop.HideTaskBar added as a startup scheduled event.

Prev Up Next

Desktop.TransIconText

Description

- Gives desktop icon text a transparent background, optionally also setting the icon to refresh automatically after a background change.

Syntax

- **Desktop.TransIconText(*autokeyword*)**

Parameters

autokeyword	(string) whether to automatically refresh icon transparency after background changes Optional. **Possible Values** *auto* — automatically refreshes transparency *[omitted]* — does not automatically refresh

Examples

- Desktop.TransIconText
  - Makes icon text transparent.

- Desktop.TransIconText("auto")
  - Makes icon text transparent & sets icon text backgrounds to automatically refresh their transparency upon background change.

Exec.Alarms

Description

- Suspend or re-activate checks for scheduled programs.

Syntax

- **Exec.Alarms**

Examples

- Exec.Alarms

Exec.Autoscroll

Description

- Starts automatic window scrolling.

Syntax

- **Exec.Autoscroll(*scrollspeed*)**

Parameters

scrollspeed	(integer) sets the number of milliseconds between auto scroll steps Optional. **Possible Values** *0* — disables automatic scrolling *scrollspeed>0* — sets scroll speed in number of milliseconds between auto scroll steps doesn't seem to do anything? *omitted* — Exec.Autoscroll command turns autoscrolling on without setting scroll speed

Mouse Commands

While the autoscroller "S" is shown, the following mouse commands apply:

Right mouse button	scroll a page
Middle mouse button	scroll 5 lines
Left mouse button	stop autoscrolling

Examples

- Exec.Autoscroll
  - Enables automatic scrolling

- Exec.Autoscroll(0)
  - Disable automatic scrolling but enable mouse commands (autoscroller "S" is still shown).

See Also

- For more information on how to control autoscrolling, see:

PowerPro Help CHM > Miscellaneous Usage Topics > Automatic scrolling with the mouse

---

PrevUp Next

Exec.AutoPress

Description

- Toggles the recognition of a window type by the mouse stop-press feature, adding it to the list of recognised window types if PowerPro does not already autopress it, or removing it if it does.

Syntax

- **Exec.AutoPress**

Examples

- Exec.AutoPress

Notes

- To use this feature, move the mouse over the window of interest and then activate the hot key.

See Also

- *PowerPro Help CHM > Miscellaneous Usage Topics > Mouse hover to click a dialog button*

Exec.BrowseRun

Description

- Shows a file open dialog, & the select file is immediately executed.

Syntax

- **Exec.BrowseRun**

Examples

- Exec.BrowseRun

Prev **Up** **Next**

Exec.CalcCalendar

Description

- Shows a PowerPro calendar dialog with 2 date/calendar fields, along with day number, week number, and the differences between the 2 dates, allowing the user to perform date calculations.

Syntax

- **Exec.CalcCalendar**

Examples

- Exec.CalcCalendar

Notes

- All values in the dialog are editable, & when changed, other fields are updated accordingly.
- Unavailable on early Win95 versions unless IE3 or later has been installed.

See Also

- *PowerPro Help CHM > Built-in Commands > *Exec > Date Calculator*

Exec.Calendar

Description

- Shows a month-view calendar with current date highlighted, along with an input field containing the current date.
- The mouse or arrow keys can be used to navigate the calendar.

Syntax

- **Exec.Calendar**

Examples

- Exec.Calendar

Notes

- Unavailable on early Windows 95 versions unless IE3 or later has been installed.

Prev Up Next

Exec.CD

Description

- Controls the audio CD player.

Syntax

- **Exec.CD(*task*)**

Parameters

task	(string) action to perform with the CD

Possible Values

"Close"	Closes the door of the default audio CD. Does not work with all types of CD drawers.
"Eject"	Ejects (opens door for) default audio CD.
"Next"	Play next track. Not working on this laptop..?
"Play"	Plays the CD from the start. Not working on this laptop..?
"Play n"	Plays the CD, starting at track number n. Not working on this laptop..?
"Previous"	Play previous track. Not working on this laptop..?
"Stop"	Stops the default audio CD.

Examples

- Exec.CD("Eject")
  - Ejects the current CD.

Notes

- Uses system audio tools...? As in, not WinAmp?
    - There are a few good WinAmp plugins available in the Yahoo! Groups PowerPro Files section.
    - I'm guessing.

See Also

- *PowerPro Help CHM > Built-in Commands > *Exec > Working with CDs*

PrevUp Next

Exec. ChangeConfiguration

Description

- Changes to configuration stored in a different pcf file; the new file path can be entered in the command.

Syntax

- **Exec.ChangeConfiguration(*pcfname*)**

Parameters

pcfname	(string) name of to the new .pcf file Optional; if omitted, file is prompted for. Path & extension are optional. If no path is included, file is searched for in PowerPro's installation directory.

Examples

- Exec.ChangeConfiguration("subbars")
  - Changes to the PowerPro "subbars" configuration.
- Exec.ChangeConfiguration("pproconf")
  - Changes back to the PowerPro's standard "pproconf.pcf" config file.
- Exec.ChangeConfiguration(?"c:\path\to\saved.pcf")
  - Changes to a config file "saved.pcf" at path given.

Exec.ClearRecent

Description

- Clears the Windows Recent folder for the current user.

Syntax

- **Exec.ClearRecent**

Examples

- Exec.ClearRecent

Notes

- Contents of the Recent folder can be displayed on the Start Menu, or found at "C:\Documents and Settings\<username>\Recent".

Prev Up Next

Exec.ClearRecentExplorer

Description

- Clears the list of recent windows shown by a Menu.Explorer call.

Syntax

- **Exec.ClearRecentExplorer**

Examples

- Exec.ClearRecentExplorer

Notes

- "Track Explorer" in PPConfig > Setup must be checked, or PowerPro will return an error.

See Also

- *PPSR > Built-in Commands > Commands > Menu > Explorer*
- *PowerPro Help CHM > Built-in Commands > *Menu > Working with Explorer windows*

Exec.CommandLine

Description

- Shows a tiny command line allowing the user to enter a command that will be run, as with Windows' Start > Run dialog.

Syntax

- **Exec.CommandLine(*ac_keywords*)**

Parameters

ac_keywords	(string) any combination of the following keywords, as a space-separated string Script calls cannot combine "urlhistory" or "urlrecent" with "history" or "=vecname". Optional. **Possible Values** *"file"* — autocomplete using file and shell folder names *"urlhistory"* — autocomplete using history URL *"urlrecent"* — autocomplete using URL most recently used *"history"* — autocomplete using the history in the dropdown of the edit box *=vecname* — where "vecname" is a vector containing strings for autocompletion

Examples

- Exec.CommandLine
  - Opens a Tiny Type & Run box.

- Exec.CommandLine("file")
  - Opens a Tiny Type & Run box that allows autocompletion with file and shell folder names.

- Exec.CommandLine("file urlhistory urlrecent")
  - Opens a Tiny Type & Run box that allows autocompletion with file and shell folder names, recently

used URL's, and URL's in the system's URL history.

- global vAutos = Vec.CreateFromWords("folder file document")
  Exec.CommandLine("=vAutos")
  - Opens a Tiny Type & Run box that allows autocompletion with the words "folder", "file", & "document".

Notes

- Only one dialog using autocomplete can be active at a time.
- The Tiny Type & Run box stays open after executing its command. It must be manually shut, possibly via its close button or context menu, or with a Window.Close_("PowerPro run") command, best not called from the same script otherwise the box will open then close again straight away.

See Also

- *PowerPro Help CHM > Built-in Commands > *Exec > Tiny Type and Run Box*
- *PowerPro Help CHM > Miscellaneous Usage Topics > Auto completion*

Exec.ContextMenu

Description

- Shows the right-click menu of a file, folder, or the window under the mouse.
- Or you can specify a file or folder to get the context menu for that file.

Syntax

- **Exec.ContextMenu(*path_or_obj*)**

Parameters

path_or_obj	(string) either: ❍ path to a file or folder ❍ name of an object from the Desktop or My Computer virtual folders Optional. If this parameter is representing an object, the same name must be used as is seen when the object is viewed in Explorer. (e.g. "My Documents")

Examples

- Exec.ContextMenu
  - ❍ Shows the context menu of the window under the mouse.
  - ❍ Note this will not work with all programs. Some that don't work include:
    - ■ Metapad
    - ■ Java-based apps
- Exec.ContextMenu(?"c:\program files\powerpro\scripts\test.powerpro")
  - ❍ Shows the context menu for the file "test.powerpro", taken as if it were being viewed in Explorer.
- Exec.ContextMenu(?"c:\program files\powerpro\scripts")
  - ❍ Shows the context menu for the file "test.powerpro", taken as if it were being viewed in Explorer.

- Exec.ContextMenu("My Computer")

Notes

- To close a PowerPro-created context menu, click the menu header (bold text bit), then click away.

- PowerPro Help recommends following this command with a
  Win.Keys("{to menu}…")
  command to select an entry from the menu, but it doesn't seem to be working..?

PrevUp Next

Exec.Disable

Description

- Disables PowerPro until the mouse is moved over a PowerPro bar, or a PowerPro hot key is pressed.

Syntax

- **Exec.Disable**

Example

- Exec.Disable

Exec.Dos

Description

- Starts Dos, runs a command line, & restarts Windows.

Syntax

- **Exec.Dos**

Examples

- Exec.Dos

Notes

- Does not work on Windows NT.
- Not working? Getting error: "Function does not work without standard explorer."

Exec.EmptyRecycleBin

Description

- Empties the recycle bin.

Syntax

- **Exec.EmptyRecycleBin**

Examples

- Exec.EmptyRecycleBin

Notes

- Unavailable on early Win95 versions unless IE4 or later has been installed.
- When entering this command via the old syntax in the PPConfig Command Edit dialog, the checkboxes can be used to control action confirmation, sound & animation.

Exec.Explorer

Description

- Opens the specified folder in a single-paned Explorer window.

Syntax

- **Exec.Explorer(*folderpath*)**

Parameters

folderpath	(string) path to the folder to show

Possible Values

a file folder	e.g. ?"c:\web"
a special folder:	Application Data Cookies Control Panel Desktop Favorites Local Settings My Computer My Documents NetHood PrintHood Recent SendTo Start Menu Templates
*	represents the current working directory of active program

Examples

- Exec.Explorer(?"c:\web")
  - Opens an Explorer window at "c:\web".

- Exec.Explorer("My Documents")
  - Opens an Explorer window at the current user's My Documents directory.

- Exec.Explorer("*")
  - Opens an Explorer window at the active program's current working directory.

See Also

- *PowerPro Help CHM > Built-in Commands > *Exec > *Exec commands*

PrevUp Next

Exec.Explorer2

Description

- Opens the specified folder in a dual-paned Explorer window.

Syntax

- **Exec.Explorer2(*folderpath*)**

Parameters

folderpath	(string) path to the folder to show **Possible Values** *as for Exec.Explorer()*

Examples

- Exec.Explorer2(?"c:\docs")
  - Opens a dual-paned Explorer window at "c:\docs".
- Exec.Explorer2("Control Panel")
  - Opens a dual-paned Explorer window at the current user's Control Panel directory.

Prev Up Next

Exec.FindFiles

Description

- Shows the Windows Find Files dialog.

Syntax

- **Exec.FindFiles**

Examples

- Exec.FindFiles
  - Opens a Find Files dialog.

- Exec.FindFiles
  Wait.For(1500)
  Win.Keys("%n*.txt")
  - Opens a Find Files dialog & sets the "Named" field to "*.txt".

- Exec.FindFiles
  Wait.For(1500)
  Win.Keys("%lc:\path;d:\p2%n*.txt")
  - Opens a Find Files dialog, setting the "Look In" field to "c:\path;d:\p2" & the "Named" field to "*.txt".

Notes

- Follow an Exec.FindFiles call with a Win.Keys() command to initialise dialog fields, as in the examples above.

Exec.FindComputer

Description

- Shows the Windows Find Computer dialog.

Syntax

- **Exec.FindComputer**

Examples

- Exec.FindComputer

Exec.HideWindow

Description

- Allows user to click a window to hide it.

Syntax

- **Exec.HideWindow**

Examples

- Exec.HideWindow

See Also

- *PowerPro Help CHM > Built-in Commands > *Exec > Hiding windows with *Exec*

PrevUp Next

Exec.HotKeys

Description

- Suspends or re-activates all or particular hotkeys.

Syntax

- **Exec.Hotkeys(*keyword[, hotkeys][, id]*)**

Parameters

keyword	(string) action to take with hotkeys **Possible Values** *"on"* *"off"* *"reverse"* — toggles hotkeys setting
hotkeys	(string) A space-separated list of hotkey names -- only names in this list will be acted on by the Exec.Hotkeys() command. Optional, if omitted, all hotkeys are affected.
id	(string) written as "<id=xxx>" where "xxx" is the ID name Optional.

Examples

- Exec.Hotkeys("on")
  - Turns hotkeys on.
- Exec.Hotkeys("off", "screentopright Semi+d")
  - Turns off the hotkeys "screentopright" & "Semi+d".
- Exec.Hotkeys("reverse")

- Toggles hotkeys on or off.

Notes

- A hotkey which runs this command will still work when hotkeys are suspended.
- If a hotkey has been disabled via the PowerPro Config dialog, it cannot be re-enabled with this function.
- Hotkey names can be found by exporting the hotkeys to an .ini file with PPConfig > Setup >Export; it is the same as the name which appears for the key in the pproconf.exe tabbed configuration dialog.
  - is the last part there (same as the "key") still correct?
- Hotkey names (ID's) can be defined a number of ways, such as in the Target Window field of PowerPro's "hotkey/mouse action" dialog with: ";<id=xxx>" -- including the semicolon & angle brackets, but not the double quotes -- or with .ini configuration files.

Prev Up Next

Exec.LogKeys

Description

- Sets keystroke logging to a file.

Syntax

- **Exec.LogKeys(*fpath*)**

Parameters

fpath	(string) either: ○ full path to target file ○ file name only ▪ The key logging file is then assumed to be in the same folder as the PowerPro configuration file. ○ either of the above, preceded by "=" character ▪ toggles key logging on/off ○ blank ("") ▪ stops key logging

Examples

- Exec.LogKeys(?"=c:\logs\keys.txt")
  - Toggles key logging to the file "keys.txt".

- Exec.LogKeys("c:/logs/keys.txt")
  Exec.ToFile("log", "Logging on: "++time)
  ; ...do stuff in here...
  Exec.ToFile("log", "Logging off: "++time)
  Exec.LogKeys("")
  - Turns on keystroke logging to the file "c:\logs\keys.txt", then uses another Exec function, Exec.ToFile(), to add activity/time details to this file whilst keystroke logging is on.
  - Note that in the Exec.ToFile() call, "log" is passed for the first parameter, the path to the open log file. The keyword "log" will only be recognised when keystroke logging is on.

Notes

- Use the *Info buttons keyword "keylog" to display an X on a button label if logging is active.

See Also

- *PowerPro Help CHM > Built-in Commands > *Exec > Logging Keystrokes*
- *PPSR > Built-in Functions > by Category > Keys > keylog, keylogfile*

PrevUp Next

Exec.Monitor

Description

- Suspend or re-activate repeated running of the special command list named "Monitor".

Syntax

- **Exec.Monitor(*keyword*)**

Parameters

keyword	(string) action to take with running of Monitor list Optional, default is "on". **Possible Values** *reverse* — toggles running of Monitor list *on* — starts running Monitor list *off* — stops running Monitor list

Examples

- Exec.Monitor("on")
- Exec.Monitor("off")
- Exec.Monitor("reverse")

Notes

- This function will only have an effect if PowerPro is configured to allow the special Monitor list (via the PPConfig > Command Lists > "Setup..." button > Special Lists dialog), & if a command list named "Monitor" exists.

See Also

- *PowerPro Help CHM > Configuring with the GUI dialogs > Command Lists > Bar and Menu "Setup" dialog > "Special Lists" tab*

PrevUp Next

Exec. MoreCommandsAsScript

Description

- This function is only relevant in the PPConfig GUI.

See Also

- *PowerPro Help CHM > Built-in Commands > *Exec > *Exec commands*

PrevUp Next

Exec.Mute

Description

- Toggles the current "Mute sounds" setting.

Syntax

- **Exec.Mute**

Examples

- Exec.Mute

Exec.NewFolder

Description

- Creates a new file folder.

Syntax

- **Exec.NewFolder(*fpath*)**

Parameters

fpath	(string) path to the new folder Optional, default is to show a PowerPro input prompt for the path. **Possible Values** *a folder path* — creates a new file folder with specified path *"" (empty)* — Shows a PowerPro input dialog that asks for a folder path, then creates a new file folder with specified path.

Examples

- Exec.NewFolder
  - Prompts for a folder path then creates a new folder as per the user input.

- Exec.NewFolder(?"f:\users\karen")
  - Creates a new folder at "f:\users\karen".

Exec.OnError

Description

- Sets where PowerPro error messages go and whether HookErrors list is run.

Syntax

- **Exec.OnError(*keywords*)**

Parameters

keywords (string) Single string comprising any combination of the keywords, separated with spaces.

Possible Values

none	Do not display any error messages.
file	Write errors messages to file ErrorLog.txt only. ▪ must be in same folder as .pcf ▪ each error creates a line of space-separated info ▪ yyyymmdd ▪ hhmmss ▪ scriptname (or *) ▪ linenumber (or 0) ▪ messagetext ▪ newlines replaced with blanks
display	Display messages on screen only.
both	Write error to file and display on screen.
hook	Run command list HookErrors whenever an error occurs.
unhook	Do not run command list HookErrors.
clear	Set _LastError_ global variable to "". Alternative form of: _LastError_=""

Examples

- Exec.OnError("none hook")
  - Stops display of error messages & turns on the command list HookErrors to be run for each PowerPro error.
- Exec.OnError("file unhook clear")
  - Sets PowerPro to write errors to the ErrorLog.txt file, turns off the HookErrors command list, & clears the global variable _LastError_.
- Exec.OnError("both")
  - Tells PowerPro to write errors to file and display them on screen.

Notes

- The global variable _LastError_ is set to the error message whenever any error occurs.
- If an error occurs while the HookErrors script is running, the script is not called again.

Prev Up Next

Exec.Plugin

Description

- Obsolete.

Exec.Print

Description

- Print a file using its associated program, such as Notepad for text files.

Syntax

- **Exec.Print(*fpath*)**

Parameters

fpath	(string) full path to the target file

Examples

- Exec.Print(?"c:\web\out.txt")

Exec.Prompt

Description

- Prompts for a Yes/No answer and sets a PowerPro flag according to the result.

Syntax

- **Exec.Prompt(*target*, *msg*)**

Parameters

target	either: ○ (integer) number of the flag to set ▪ Prompt will have 2 buttons: Yes & No. ○ (string) a variable name ▪ Prompt will have 3 buttons: Yes, No & Cancel.
msg	(string) text to appear in the prompt dialog

Examples

- Exec.Prompt(14, "Do you like chocolate?")
  Win.Debug(pproflag(14))
  - Shows a message box asking "Do you like chocolate?", with 2 buttons, Yes & No, then sets flag 14 to the result.

- local ans
  Exec.Prompt("ans", "Do you like chocolate?")
  Win.Debug(ans)
  - Shows a message box asking "Do you like chocolate?", with 3 buttons, Yes, No & Cancel, and sets variable ans to the result.

Exec.QuoteEscape

Description

- Sets whether single quote is an escape character in strings in expressions.

Syntax

- **Exec.QuoteEscape(*keyword*)**

Parameters

keyword	(string) action to take

Possible Values

"on"	
"off"	
"reverse"	

Examples

- Exec.QuoteEscape("off")

Notes

- Only works if Setup >Advanced >Characters "Escape character" is set to single quote.

Prev Up Next

Exec. RefreshEnvironment

Description

- Refreshes all environment variables from registry; does not delete variables which are deleted.

Syntax

- **Exec.RefreshEnvironment**

Examples

- Exec.RefreshEnvironment

Notes

- Windows NT or later only.

Prev Up Next

Exec.RestoreLastMin

Description

- Restores the last minimized window.

Syntax

- **Exec.RestoreLastMin**

Examples

- Exec.RestoreLastMin

Prev **Up** **Next**

Exec.SchedulerAdd

Description

- Adds a new scheduled event.

Syntax

- **Exec.SchedulerAdd(*evdate, evtime, action*)**

Parameters

evdate	(integer) date of the event **Possible Values** *yyyymmdd* — exact date *0 < evdate < 1000* — number of days from now (zero for today)
evtime	(integer) time of the event **Possible Values** *hhmm* — time in 24 hour clock time *+hhmm* — number of hours and minutes from now
action	(string) either: ❍ a PowerPro command ❍ text, which will be set as a Win.Message() event.

Examples

- Exec.SchedulerAdd(20061225, 0900, "Merry Christmas")
  - ❍ Creates a new scheduled event to show a PowerPro message window with the message "Merry Christmas" at 9am on Christmas Day, 2006.

- Exec.SchedulerAdd(0, +100, "c:\program files\myprog\myprog.exe")
  - ❍ Creates a new scheduled event to run myprog.exe one hour from now.

Notes

- Exec.SchedulerAdd cannot be used in a script called from a scheduled alarm, unless you precede the Exec.SchedulerAdd() call by Wait.For(1000).
- The parameters evdate & evtime must each be a single number without colons, slashes, or other delimiters.

PrevUp Next

Exec.Scroll

Description

- Begins mouse-scrolling the window under the mouse, as if a middle-click had been pressed.

Syntax

- **Exec.Scroll**

Examples

- Exec.Scroll

Notes

- To stop scrolling, left-click anywhere on the screen.
- This function is equivalent to Exec.ScrollWithMouse.

See Also

- *PowerPro Help CHM > Miscellaneous Usage Topics > Manual scrolling with the mouse*

Exec.ScrollWindow

Description

- Scrolls the window under the mouse.

Syntax

- **Exec.ScrollWindow(*numRows*)**

Parameters

numRows	(integer) number of rolls to scroll

Possible Values

positive integer	window scrolls down
negative integer	window scrolls up

Examples

- Exec.ScrollWindow(5)
  - Scrolls down 5 lines in the window under the mouse.

- Exec.ScrollWindow(-5)
  - Scrolls up 5 lines in the window under the mouse.

Exec.ScrollWithMouse

Description

- Begins mouse-scrolling the window under the mouse, as if a middle-click had been pressed.

Syntax

- **Exec.ScrollWithMouse**

Examples

- Exec.ScrollWithMouse

Notes

- To stop scrolling, left-click anywhere on the screen.
- This function is equivalent to Exec.Scroll.

See Also

- *PowerPro Help CHM > Miscellaneous Usage Topics > Manual scrolling with the mouse*

Prev Up Next

Exec.SetEnv

Description

- Set environment variable to provided text. Use the env(...) function in expressions to read environment variables.

Syntax

- **Exec.SetEnv(*varname*, *text*)**

Parameters

varname	(string) name of the Windows Environment variable to set
text	(string) new value for varname

Examples

- Exec.SetEnv("testEnvVar", "PowerPro is cool")
  Win.Debug(env("testEnvVar"))
  - Creates an environment variable "testEnvVar" & sets it to the string "PowerPro is cool", or resets the variable to that string if it already existed. The variable's value is then debugged.

Notes

- If the Environment variable already exists, any previous value will be overwritten by this function.

See Also

- *PPSR > Built-in Functions > by Category > System > env*

Exec. StandardConfiguration

Description

- Checks whether PPConfig > Setup >Advanced >"Use standard configuration" is checked & if not, outputs an error message & stops all scripts.

Syntax

- **Exec.StandardConfiguration**

Examples

- Exec.StandardConfiguration

PrevUp Next

Exec.Tile

Description

- Tiles the visible windows on the screen.

Syntax

- **Exec.Tile**

Examples

- Exec.Tile

Prev Up Next

Exec.ToFile

Description

- Writes a single line of text to a log file, including a line break at the end.

Syntax

- **Exec.ToFile(*fpath*, *text*)**

Parameters

fpath	(string) path to the target file **Possible Values** *a file path* *"log"* — The currently open log file, as specified by an Exec.LogKeys () call.
text	(string) the text to write to the file

Examples

- Exec.ToFile(?"c:\web\out.txt", "some text to write")
- Exec.ToFile(?"c:\web\out.txt", date++":"++time)

Notes

- Be aware that Exec.ToFile() will only write a single line of text -- and that means no line breaks within that string, either. (That will draw errors, too, if the string begins with a line break.)

See Also

- Use the File plugin for file input and output with multi-lined strings. For more information, see:
  - *PPSR > Plugins > The Plugins > File*
- the second example in Exec.LogKeys in this same section

PrevUp Next

Exec.VolumeAll

Description

- Set the volume for all types of playback.

Syntax

- **Exec.VolumeAll(*level*)**

Parameters

level	(integer) sets the volume level

Possible Values

0<level<255	a value between 0 (mute) and 255 (loudest)
+level	increases the volume by the amount of level
-level	decreases the volume by the amount of level

Examples

- Exec.VolumeAll(120)
  - Sets the system volume to 120.
- Exec.VolumeAll(-120)
  - Decreases the system volume by 120.

Prev Up Next

Exec.VolumeWav

Description

- Sets the volume for .wav files.
- Enter number 0 (mute) to 15 (loudest). Use + or - in front of number to adjust relative to current setting.

Syntax

- **Exec.VolumeWav(*level*)**

Parameters

level	(integer) sets the volume level

Possible Values

0<level<15	a value between 0 (mute) and 15 (loudest)
+level	increases the volume by the amount of level
-level	decreases the volume by the amount of level

Examples

- Exec.VolumeWav(10)
  - Sets the volume for .wav files to 10.
- Exec.VolumeWav(-10)
  - Decreases the volume of .wav files by 10.

Prev Up Next

Exec.WindowInfo

Description

- Displays/Hides a small PP window with information about the window under the mouse.
- Information includes the mouse screen position, plus the size, position, caption, class, and exe name of the window under the mouse.

Syntax

- **Exec.WindowInfo**

Information Displayed

- 1. mouse screen coordinates
  - Absolute
    - point (0,0) is top-left of screen
  - Relative
    - point (0,0) is top-left of window under mouse
- 2. window coordinates of window under mouse
  - (left,top) - (right,bottom)
- 3. window size info
  - main window size
    - width x height of total window
  - client window size
    - width x height of window area excluding the border, caption, menu bar, tool bars, and status bar
  - aspect ratio
    - width of client window/height of client window
- 4. window caption
- 5. main window info
  - handle in hex

    - class
    - exe name

- 6. child window info
    - handle in hex
    - class

Examples

- Exec.WindowInfo

Notes

- The Exec.Window commands acts as a toggle for the info window; to hide it, execute the command again.
- If the mouse cursor is over an Edit box, the contents of that box are shown as the caption. This can be useful to see password fields.

PrevUp Next

File.CommandRandom

Description

- Runs a command with a randomly selected file as a parameter.

Syntax

- **File.CommandRandom(*commandpath, filepath, args, hide*)**

Parameters

commandpath	(string) path to the command file or program
filepath	(string) file path with wildcards
args	(mixed) arguments to use with the command Optional.
hide	(boolean) *1 or "hide"* — hides any output from the command *0 or "" (empty)* — shows any output from the command Optional, default is 0.

Examples

- File.CommandRandom(?"c:\program files\bat\exec.bat", ;;+
  ?"c:\random*.*", date, time, 1)
  - Uses a file selected randomly from "c:\random*.*" as the 1st parameter in the call to exec.bat, with the current date as the 2nd parameter & current time as the 3rd, running the command invisibly.

- File.CommandRandom(?"c:\utils\converter.exe", ;;+
  ?"c:\wallpapers*.jpg", ?"c:\store\temppaper1.bmp")
  - Passes a random .jpg file from "c:\wallpapers" to the converter.exe program, along with another parameter "c:\store\temppaper1.bmp", displaying any program windows on screen.

See Also

- *PowerPro Help CHM > Built-in Commands > *File > Working with a randomly selected file*

PrevUp Next

File.Copy

Description

- Makes a copy of a file to the specified path.

Syntax

- **File.Copy(*source, destination*)**

Parameters

source	(string) full path to the file to be copied
destination	(string) path to the destination file If file name not included in path, source file name is used.

Examples

- File.Copy(?"c:\mypath\in.txt", ?"c:\output\out.txt")
  - Copies "c:\mypath\in.txt" to "c:\output\out.txt".

- File.Copy(?"c:\docs*.txt", ?"c:\backup\")
  - Use the "*" wildcard to that specify all text files in "c:\docs" are to be copied to "c:\backup".

- File.Copy(?"c:\mydocs\file?.txt", ?"c:\logs")
  - Uses the "?" wildcard to specify that all text files with a file name consisting of the word "file" plus one more character & the extension should be copied to folder "c:\logs".

File.CopyRandom

Description

- Copies a randomly selected file to a specified file path.

Syntax

- **File.CopyRandom(*filepath*, *outfile*)**

Parameters

filepath	(string) path with wildcards
outfile	(string) destination for file copy

Examples

- File.CopyRandom(?"c:\pictures*.bmp", ?"c:\logo.sys")
  - Copies a random .bmp file from the directory "c:\pictures" over the Windows startup screen file. (Windows 95/98/ME)

See Also

- *PowerPro Help CHM > Built-in Commands > *File > Working with a randomly selected file*

File.Delete

Description

- Deletes a file, or group of files.

Syntax

- **File.Delete(*filepath*)**

Parameters

filepath	(string) path to the file to delete Use wildcards to define a group of files.

Examples

- File.Delete(?"c:\temp\large.tmp")
  - Deletes the file "large.tmp" in "c:\temp".
- File.Delete(?"c:\temp*.tmp")
  - Deletes all .tmp files in "c:\temp".
- File.Delete(?"c:\temp*")
  - Deletes all files in "c:\temp".

File.DeleteNoRecycle

Description

- Completely deletes a file or group of files, bypassing the recycle bin.

Syntax

- **File.DeleteNoRecycle(*filepath*)**

Parameters

filepath	(string) path to the file to delete Use wildcards to define a group of files.

Examples

- File.DeleteNoRecycle(?"c:\temp\large.tmp")
  - Deletes the file "large.tmp" in "c:\temp" without putting it in the recycle bin.

- File.DeleteNoRecycle(?"c:\temp*.tmp")
  - Deletes all .tmp files in "c:\temp" without putting them in the recycle bin.

File.DeleteOld

Description

- Deletes files in folder older than specified number of days

Syntax

- **File.DeleteOld(*numdays*, *folderpath*)**

Parameters

numdays	(integer) number of days file must be older than
folderpath	(string) path to the folder containing the files to delete Use wildcards to define a subset of files.

Examples

- File.DeleteOld(5, ?"c:\temp")
  - Deletes all files in "c:\temp" more than 5 days old.
- File.DeleteOld(5, ?"c:\temp*.tmp")
  - Deletes all .tmp files in "c:\temp" more than 2 days old.
- File.DeleteOld(3, "Recent")
  - Deletes all files in the Recent folder that are older than 3 days.

File.ExtChange

Description

- Change, add, or remove a file extension.

Syntax

- **File.ExtChange(*fpath, newext*)**

Parameters

fpath	(string) path to the file to change **Possible Values** *file character: |* — Used in file selection context menu commands to indicate the selected file. For more informaton, see: PPSR > Built-in Functions > by Category > File, Disk > "|" *path expression*
newext	(string) new extension for fpath

Examples

- File.ExtChange("|", "txt")
  - If this command was placed in the special command list Context, it could change the extension of the selected file in an Explorer-type window to ".txt".

File.Move

Description

- Moves a file or files to another location, effectively renaming it to the new path.

Syntax

- **File.Move(*source*, *destination*)**

Parameters

source	(string) full path to the file to be moved Use wildcards to move groups of files.
destination	(string) path to the new file location If destination does not included a file name, source file name is used.

Examples

- File.Move(?"c:\mypath\in.txt", ?"c:\output\out.txt")
  - Moves "c:\mypath\in.txt" to "c:\output\out.txt".

- File.Move(?"c:\docs*.txt", ?"c:\backup\")
  - Use the "*" wildcard to that specify all text files in "c:\docs" are to be moved to "c:\backup".

- File.Move(?"c:\mydocs\file?.txt", ?"c:\logs")
  - Uses the "?" wildcard to specify that all text files with a file name "file" plus one character & extension .txt should be moved to folder "c:\logs".

Notes

- This function is the same as File.Rename().

Prev Up Next

File.Rename

Description

- Renames a file or files, effectively moving it to the new path.

Syntax

- **File.Rename(*source, destination*)**

Parameters

source	(string) full path to the file/s to be renamed Use wildcards to rename groups of files.
destination	(string) path to the new file/s location If destination does not included a file name, the source file's name is used.

Examples

- File.Rename(?"c:\mypath\in.txt", ?"c:\output\out.txt")
  - Renames "c:\mypath\in.txt" to "c:\output\out.txt".

- File.Rename(?"c:\docs*.txt", ?"c:\backup\")
  - Use the "*" wildcard to that specify all text files in "c:\docs" are to be renamed (i.e. moved) to folder "c:\backup".

- File.Rename(?"c:\mydocs\file?.txt", ?"c:\logs")
  - Uses the "?" wildcard to specify that all text files with a file name "file" plus one character & the extension .txt, should be moved to folder "c:\logs".

Notes

- This function is the same as File.Move().

Prev Up Next

File.RunRandom

Description

- Selects a file at random from a folder & runs it using the program associated with the file extension.

Syntax

- **File.RunRandom(*fpath*)**

Parameters

fpath	(string) path with wildcards

Examples

- File.RunRandom(?"c:\sounds*.wav")
  - Plays a random .wav file from "c:\sounds".

See Also

- *PowerPro Help CHM > Built-in Commands > *File > Working with a randomly selected file*

Format.BarVerticalLine

Description

- Draws a vertical line on a bar.

Syntax

- ***Format BarVerticalLine**

Examples

- *Format BarVerticalLine

Format.Context

Description

- Starts a menu/bar section , or shows a whole bar, depending on the active window.

Syntax

- ***Format Context expr**

Parameters

expr	(mixed) caption list specifying the context for the bar/menu to be displayed

Examples

- *Format Context =metapad
  - Used as the first command of a bar, this would make the bar be displayed only when a Metapad window is the activewindow. A closing *Format EndContext command is not required in this situation.
  - Used within another menu/bar, this would start a contextual section that is only shown when Metapad is active. In this case, it must be followed by a *Format EndContext command.

PrevUp Next

Format.ContextIf

Description

- Starts a menu/bar section , or shows a whole bar, depending on the active window, where the context is given by an expression.

Syntax

- ***Format ContextIf expr**

Parameters

expr	(mixed) caption list expression that must be evaluated to true for the bar/menu to be displayed

Examples

- *Format ContextIf (modem or a>0)
  - Used as the first command of a bar, this would make the bar be displayed only when the dialup modem is connected, or variable a is greater than 0. A closing *Format EndContext command is not required in this situation.
  - Used within another menu/bar, this would start a contextual section that is only shown when the dialup modem is connected, or variable a is greater than 0. In this case, it must be followed by a *Format EndContext command.

Format.Drag

Description

- Gives click-drag functionality to any bar button assigned this command, whereby click-dragging that button will move the bar.

Syntax

- ***Format Drag**

Examples

- *Format Drag

PrevUp Next

Format.EndContext

Description

- Ends a section started with a *Format Context or *Format ContextIf command.

Syntax

- ***Format EndContext**

Examples

- *Format EndContext

Format.EndSubMenu

Description

- Ends a submenu in a menu.

Syntax

- ***Format EndSubMenu**

Examples

- *Format EndSubMenu

Notes

- A menu is started with a *Format StartSubMenu command.
- Menus may be nested.

Format.Item

Description

- Obsolete.
- Use CL functions instead, see:
  - *PPSR > CL Functions*

PrevUp Next

Format.MaxColumn

Description

- Sets the maximum number of entries per column in a menu.

Syntax

- ***Format MaxColumn n**

Parameters

n	(number) maximum number of entries per column

Examples

- *Format MaxColumn 20
  - Sets the maximum number of menu items per column to 20.

Notes

- May not be effective with embedded submenus, in which case use explicit *Format NewColumn commands.

Format.NewBarRow

Description

- Starts a new row in a bar.

Syntax

- ***Format NewBarRow**

Examples

- *Format NewBarRow

PrevUp Next

Format. NewBarRowLine

Description

- Starts a new row in a bar & draws a separator line.

Syntax

- ***Format NewBarRowLine**

Examples

- *Format NewBarRowLine

PrevUp Next

Format.NewColumn

Description

- Starts a new column in a menu.

Syntax

- ***Format NewColumn**

Examples

- *Format NewColumn

PrevUp Next

Format. NewColumnLine

Description

- Starts a new column with a vertical separator line.

Syntax

- ***Format NewColumnLine**

Examples

- *Format NewColumnLine

Format.Separator

Description

- Draws a vertical separator on a bar or menu.

Syntax

- ***Format Separator**

Examples

- *Format Separator

Prev Up Next

Format.StartSubMenu

Description

- Starts a submenu in a menu.

Syntax

- ***Format StartSubMenu**

Examples

- *Format StartSubMenu

Notes

- Must be closed with *Format EndSubMenu

PrevUp Next

Menu.Explorer

Description

- Shows a menu of folders recently accessed with Explorer.

Syntax

- **Menu.Explorer**

Examples

- Menu.Explorer

Notes

- "Track Explorer" on the Setup dialog must be checked, or PowerPro will return an error.

Prev**Up** **Next**

Menu.Folder

Description

- Shows a menu of files from a folder.

Syntax

- **Menu.Folder(*folderPath, sFormat*)**

Parameters

folderPath (string) path to the base folder

Optional, defaults to the logged in user's "Start Menu" directory.

Possible Values

- full path to a folder
- the name of any folder in the currently logged in user's profile folder, such as:
  - Application Data
  - Cookies
  - Desktop
  - Favorites
  - Local Settings
  - My Documents
  - NetHood
  - PrintHood
  - Recent
  - SendTo
  - Start Menu
  - Templates
- any of the following Keywords:

AllPrograms	Menu of Start Menu Programs folders for All Users profile.
Allprograms FFF	Menu of Start Menu Programs folder FFF for All Users profile.

NT4 only.

AllStartMenu	Start menu for All Users profile NT4 only.
AllDesktop	Desktop for All Users profile. NT4 only.
personal	Shortcut for current user's Favorites folder
programs	Programs folder from Start Menu for current user. NT4 only.
programs FFF	menu of Programs folder FFF (e.g. Accessories) NT4 only.

sFormat

(string) any combination of the following keywords indicating desired menu format

Optional.

Keywords

**all*	Execute all commands/programs in folderPath, rather than displaying a menu.
**allclose*	Close all commands/programs in folderPath, rather than displaying a menu.
**allcloseforce*	Force closed all commands/programs in folderPath, rather than displaying a menu.
autocol n	Automatically starts a new column every n+1 entries. Applies to top level menu only.
autocolall n	Automatically starts a new column every n+1 entries. Applies to top level menu & submenus.
autosoftcol n	Automatically starts a new column every n+1 entries, without including a bar between the columns.

	Applies to top level menu only.
autosoftcolall n	Automatically starts a new column every n+1 entries, without including a bar between the columns. Applies to top level menu and submenus.
back	Bar or menu takes the default background defined in PPCD > Command Lists > Setup.
center	Centers menu on screen.
cmd 'cmdline' 'params'	Defines a command line & parameters to be executed with the selected file, where: *'cmdline'* — command to be executed, in single quotes *'params'* — parameters for 'cmdline', in single quotes Use the _file_ variable or a "|" to indicate where the selected file is to be placed in the command line. The command line & parameter strings that appear after the "cmd" keyword can optionally be enclosed in nested single quotes, unless the strings contain spaces, in which case single quotes must be used. ***Example*** ▪ Menu.Folder("desktop", "cmd 'c:\program files\2x \2xExplorer' '|*'")
empty	Empty folders will be included in the menu (normally, they are excluded).
exclude \"folders_list\"	Excludes all folders listed in folders_list, a comma-separated list of folders. ***Examples***

- Menu.Folder(?"c:\temp", "exclude \"c:\\temp\\store, c:\\temp\\logs\"")
  - Shows a menu of the "c:\temp" directory, excluding the "store" and "logs" folders.
- Menu.Folder(?"c:\temp", ?#exclude "c:\temp\store, c:\temp\logs"#)
  - Shows a menu of the "c:\temp" directory, excluding the "store" and "logs" folders.

Doesn't work with other exclusion-type keywords:

nofiles	
nofolders	
nosubdir	
nosubmenu	

explorer	Adds a menu entry with label "Explore" at the top of all submenus, a shortcut to open a single-paned Explorer window.
	
	The entry can be:
	- left-clicked
	  - to open a single-paned Explorer window on the selected directory
	- right-clicked
	  - to show a Menu.Folder() menu for that folder, useful if files are not being displayed in the current menu.
	  - Am getting an error when right-clicking explorer entry:
	
	Cannot access context menu for:
	c:\docs & settings\username\start menu\programs\
	Error occurred near line 1 which contained:
	Menu.Folder("programs", "explorer fileman

	\"open in explorer\" ")

explorer2	Adds a menu entry with label "Explore" at the top of all submenus, a shortcut to open a double-paned Explorer window.
	
	The entry can be:
	▪ left-clicked
	  ▪ to open a double-paned Explorer window at the selected directory
	▪ right-clicked
	  ▪ to show a Menu.Folder() menu for that folder, useful if files are not being displayed in the current menu.

fileman \"lbl\"	Where lbl = label for the optional top item of submenus, created when the "explorer" keyword is included. The "explorer" keyword must also be included in sFormat whenever "fileman" is used. ***Example*** ▪ Menu.Folder("programs", "explorer fileman \"Open in Explorer...\" ")
folderback	Adds a "Back" entry to the top of the menu, to go back to previous folder. Only useful when NoSubDir is specified.
folderdots	Adds "..." to folder names. Useful to highlight folders when NoSubDir is specified & icons are not used in menus.
foldershorcut	Expands all folder shortcuts in the menu.
foldershorcuts2	Expands foldershortcuts with names ending in "_x".
folderstart	Sorts menu entries so that folders are at the start.
noicons	Omits menu icons.

	Only works if the Menu.Folder() command is not embedded in another menu.
maxtext n	Limits text labels to n characters.
mne	Powerpro assigns single character menu mnemonics to the first 36 items on the main menu, allowing them to be easily selected with the keyboard.
noext	File extensions are removed from menu item labels.
nofiles	Files are not displayed, only folders. Useful with the "explorer" option to traverse large folder trees quickly.
nofolders	Omit all subfolders.
nosort	Menu items will not be sorted.
nosubdir	Dynamic subdirectories (that appear when the parent items are moused over) are not included. The names of subdirectories are still shown, & if selected, a Menu.Folder() is shown for that subdirectory. Doesn't work with "exclude" keyword.
nosubmenu	All files from all subdirectories will be listed in the main menu. No folders are shown.
offset n1 n2	Positions the menu offset n1 characters to the right, & n2 characters below the mouse cursor. n1 & n2 can be negative.
omit	Deletes the phrases in the "Omit strings..." edit box on the PProConf > Command Lists > Setup > All Bars and Menus > Config tab. "omit" is applied before "maxtext n", if both are specified.
sortext	Menu items are sort by file extension.
sorttime	Sorts most recently change files to the start of the menu.

tipall	Shows up to 6 lines of the file contents in the tool tip. Some file types (text-based, such as .txt, .html, .ini) work better than others (such as .exe) with this keyword.
tiptext	Shows up to 6 lines of .txt file contents on tool tip.
tool tips	Shows tool tips, regardless of PProConf setting on Tool Tip Setup dialog.
under	Centers the menu under the mouse.
Zero \"vars_list\"	vars_list = space-separated list of variables to be set to 0 before the files are processed.

Examples

- Menu.Folder("c:/jane/docs")
  - Creates a menu of files from folder "c:\jane\docs".
- Menu.Folder("c:/jane/docs", "nofolders")
- Menu.Folder("programs", "explorer fileman \"open in explorer\" ")
- Menu.Folder(?"c:\my pics", "folderback folderdots folderstart nosubdir center tipall")

Notes

- Use wildcard matches to limit the files shown.

See Also

- *PowerPro Help CHM > Built-in Commands > *Menu > *Menu Folder*
- *PowerPro Help CHM > Built-in Commands > *Menu > *Menu Folder formatting*
- *PowerPro Help CHM > Built-in Commands > *Menu > Special folders for *Menu Folder*
- *PowerPro Help CHM > Built-in Commands > *Menu > Entering format information for *Menu Folder Command*

PrevUp Next

Menu.Recent

Description

- Shows a menu of recently executed windows (and/or commands), & the selected item is re-executed.

Syntax

- **Menu.RecentCommands**

Notes

- "Track windows for *Menu RecentCommands" in "PProConf > Command Lists > Setup > All Menus" must be checked.
- Windows shown in the list are the ones most recently executed by any means, not just via PowerPro.

Examples

- Menu.RecentCommands

Menu.StartMenu

Description

- Shows the Windows start menu at the mouse cursor.

Syntax

- **Menu.StartMenu**

Examples

- Menu.StartMenu

Notes

- Windows will initially show the menu in the standard location at lower left of screen, before the menu will appear to jump to the cursor. This repositioning of the menu is unavoidable at present.

See Also

- *PPSR > Built-in Commands > *Menu > Folder*

Prev Up Next

Menu.Show

Description

- Shows a command list as a menu, positioned according to specified keyword.

Syntax

- **Menu.Show(*clname, position*)**

Parameters

clname	(string) name of the command list to use for the menu
position	(string) one of the following position keywords:

Possible Values

centerUnderMouse	Centers the menu under the mouse.
centerScreen	Centers the menu on the screen.
offset n1 n2	Shows the menu positioned n1 pixels to the left & n2 pixels above the mouse. n1 & n2 can be negative, which would position the menu to the right & below the cursor respectively. Note ▪ If the config setting "Force cursor over new opened menus" (in PProConf > Command Lists > Setup > All Menus) is checked, this setting will not have any effect.
screen n1 n2	Shows the menu at screen position (n1, n2), where: *n1 = pixels from left* *n2 = pixels from top*

horButton	Shows the menu aligned under/above the last button pressed on a horizontal bar.
horButtonCenter	Shows the menu centered under/above the last button pressed on a horizontal bar.
verButton	Shows the menu aligned left/right of the last button pressed on a horizontal bar.

Examples

- Menu.Show("Locations", "offset 20 20")
  - Shows the command list named "Locations", positioned 20px above & 20px to the left of the mouse.

- Menu.Show("Locations", "centerScreen")
  - Shows the command list named "Locations", positioned in the centre of the screen.

Notes

- Use the Format command to add submenus and separators to a menu.

- The position keywords "horbutton" and "verbutton" perform the same alignment as a Menu.ShowAtButton() command, however, by using Menu.Show (), you can more precisely specify the menu's bar alignment.

See Also

- *PowerPro Help CHM > Built-in Commands > *Menu > Showing menus*

PrevUp Next

Menu.ShowAtButton

Description

- Meant for use on bar buttons, this command shows a menu aligned with the button used to display it.

Syntax

- **Menu.ShowAtButton(*clname*)**

Parameters

clname	(string) name of the command list to use for the menu

Examples

- Menu.ShowAtButton("Locations")
  - This command could be used with a button labelled "Locations", which would open the Locations command list as a menu aligned with this button.

Notes

- For horizontal bars, the menu is shown below or above the button, depending on which half of the screen the bar is in.
- For vertical bars, the menu is shown to the left or right of the button, depending on which half of the screen the bar is in.

See Also

- Menu.Show()
  - *PPSR > Built-in Commands > Menu > Show*

Menu.ShowAtCursor

Description

- Shows a command list as a menu at a text cursor, if present.

Syntax

- **Menu.ShowAtCursor(*clname*)**

Parameters

clname	(string) name of the command list to use for the menu

Examples

- Menu.ShowAtCursor("Snippets")
  - Shows the command list "Snippets" as a menu aligned with the current document's text prompt.

Notes

- not working..?

See Also

- Menu.Show()
  - *PPSR > Built-in Commands > Menu > Show*

Menu.ShowFile

Description

- Shows the contents of a file as a menu.

Syntax

- **Menu.ShowFile(*fpath*)**

Parameters

fpath	(string) path to the file containing the menu

Examples

- Menu.ShowFile(?"c:\mine\stuff.txt")
  - Shows the file "stuff.txt" as a menu.

Notes

- Use *Format commands in the file to format the menu, create submenus, add separators & conditionally include items.

- The following Properties Keywords may be used within the menu file for formatting:

*back **	Sets menu background to use desktop wallpaper.
back "fpath"	fpath = path (in quotes) to .bmp for menu background
back r g b	r g b = red green blue components of menu background colour Each value is between 0 & 255.
fontbold	Sets menu font to bold.
fontitalics	Sets menu font to italics.
fontname fn	fn = font name Use double quotes if fn contains spaces. Must be specified for other font keywords to be actioned.
fontsize n	n = pixel size of font

fontwidth n	n = boldness of font, as a number 0 < n < 255
iconsize	Makes icons included & sets icon size. Must be specified for icons to be shown.
maxtext n	n = maximum length of menu labels 0 < n < 255
text r g b	r g b = red, green, blue components of text colour Each value is between 0 & 255.
tooltips	Shows tool tips for menu items.

See Also

- Alternative command:
  - *PPSR > Built-in Functions > by Category > Files > filemenu()*
- More information on formatting file menus:
  - *PowerPro Help CHM > Built-in Commands > *Menu > File menus*
  - *PowerPro Help CHM > Built-in Commands > *Menu > Properties in file menus*
- The {filemenu fpath} argument in Win.Keys():
  - *PPSR > Plugins > The Plugins > Win > Keys*

Menu.Tray

Description

- Shows a menu of all current tray icons.

Syntax

- **Menu.Tray**

Examples

- Menu.Tray

See Also

- *PowerPro Help CHM > Built-in Commands > *TrayIcon > Show Tray Icons in Active Bar or Menu*

Menu.UnderMouse

Description

- Show the menu of the window under the mouse.
- Windows 95/98 only.

Syntax

- **Menu.UnderMouse**

Examples

- Menu.UnderMouse

Notes

- Must be assigned to a hot key or as part of menu shown by hot key.
- Only shows standard menus, not toolbars like the menu tool bar in Explorer.

PrevUp Next

Note.CloseCategory

Description

- Closes all notes open from specified category.

Syntax

- **Note.CloseCategory(*cat*)**

Parameters

cat	(string) note category

Possible Values

*	all categories
a single category name	

Examples

- Note.CloseCategory("work")
  - Closes all notes in category "work".

Note. DeleteOpenCategory

Description

- Deletes notes opened from specified category.

Syntax

- **Note.DeleteOpenCategory(*cat*)**

Parameters

cat	(string) note category

Possible Values

*	all categories
a single category name	

Examples

- Note.DeleteOpenCategory("clipstore")
  - Deletes all notes from category "clipstore" that are open.

Note.Open

Description

- Creates a new note or opens an existing note (or notes) & returns the note's handle.

Syntax

- **Note.Open(*fpath, cat_expr*)**

Parameters

fpath	(string) full path to a note, including .ppronote extension Optional, if omitted, a new note is created. If specified, the note must already exist.
cat_expr	(string) category string, in format "cat catname" where catname=category name Optional, but it must be the 2nd parameter, & the 1st parameter must be an empty string (otherwise the 2nd parameter is ignored).

Return Value

- (integer) the note's handle

Examples

- Note.Open
  - Opens a new note.
- global nh = Note.Open
  - Opens a new note & assigns its handle to the global variable nh.
- Note.Open(pprofolder++?"notes\dev\stuff.ppronote")
  - Opens the note "stuff.ppronote".
- Note.Open("", "cat work")

- Opens a new note in category "work".

- Note.Open("","cat "++date)
  - Opens a new note with its category set to today's date.

PrevUp Next

Note.OpenCategory

Description

- Opens all notes in specified category.

Syntax

- **Note.OpenCategory(*cat*)**

Parameters

cat	(string) note category

Possible Values

*	all categories
a single category name	

Examples

- Note.OpenCategory("work")
  - Opens all notes in category "work".

Note.OpenMenu

Description

- Shows a menu of all open notes, & brings the selected note to the top.

Syntax

- **Note.OpenMenu**

Examples

- Note.OpenMenu

PrevUp **Next**

Note. OpenOneFromMenu

Description

- Shows a menu of notes from specified category & opens the selected note.

Syntax

- **Note.OpenOneFromMenu(*cat*)**

Parameters

cat	(string) note category

Possible Values

*	all categories
a single category name	

Examples

- Note.OpenOneFromMenu("lists")
  - Shows a menu of all notes in category "lists", then opens the chosen note.

Prev Up Next

Note.OpenToday

Description

- Shows notes with date categories today or before

Syntax

- **Note.OpenToday**

Examples

- Note.OpenToday

Notes

- not working..? does nothing.

Prev Up Next

Note.ShowHideOpen

Description

- Toggles hidden/shown state of open notes.

Syntax

- **Note.ShowHideOpen(*cat*)**

Parameters

cat	(string) note category

Possible Values

*	all categories
a single category name	

Examples

- Note.ShowHideOpen("*")
  - Toggles visibility of all open notes.

- Note.ShowHideOpen("work")
  - Toggles visibility of all notes open from category "work".

Note.ShowOpen

Description

- Shows all hidden notes from a category.

Syntax

- **Note.ShowOpen(*cat*)**

Parameters

cat	(string) note category

Possible Values

*	all categories
a single category name	

Examples

- Note.ShowOpen("work")
  - Shows all notes in category "work" that are open but hidden.

ScreenSaver.Change

Description

- Changes to a random screen saver file (.scr file) in same folder.

Syntax

- **ScreenSaver.Change(*folder_path*)**

Parameters

folder_path	(string) full path to folder containing screen saver files If a filename is included in the path, it is ignored. Optional. ❍ If omitted, PowerPro appears to search the whole hard drive..?

Examples

- ScreenSaver.Change(?"c:\pics\savers")
  ScreenSaver.Start
  - ❍ Changes to a new screensaver from the folder "c:\pix\savers", then starts it.

Notes

- PowerPro Help CHM says ScreenSaver.Change() picks files either at random or sequentially, but I am unable to get sequential savers (assuming that sequentially as in alphabetically).

ScreenSaver. ChangeRestart

Description

- Sets, clears, or reverses the "Restart running saver if changed" setting in PowerPro Config > GUI Control dialog.

Syntax

- **ScreenSaver.ChangeRestart(*state*)**

Parameters

state	(string) action to take with setting

Possible Values

clear	clears the setting
set	checks the setting's checkbox
reverse	toggles the current setting state

Examples

- ScreenSaver.ChangeRestart("set")
  - Ticks the checkbox next to the setting "Restart running saver if changed" in PProConf.
- ScreenSaver.ChangeRestart("reverse")
  - Reverses the setting "Restart running saver if changed" in PProConf.

ScreenSaver. ChangeTimeout

Description

- Changes saver timeout (time before saver kicks in) to the specied value.

Syntax

- **ScreenSaver.ChangeTimeout(*mins*)**

Parameters

mins	(integer) new timeout value in minutes

Examples

- ScreenSaver.ChangeTimeout(10)

PrevUp Next

ScreenSaver.ChangeTo

Description

- Changes screen saver to file specified.

Syntax

- **ScreenSaver.ChangeTo(*fpath*)**

Parameters

fpath	(string) full path to the screensaver file

Examples

- ScreenSaver.ChangeTo(?"c:\windows\system32\sspipes.scr")
  - Changes screen saver to "sspipes.scr".

Notes

- not working..?

PrevUp Next

ScreenSaver.Disable

Description

- Disables the screen saver.

Syntax

- **ScreenSaver.Disable**

Examples

- ScreenSaver.Disable

Prev Up Next

ScreenSaver.Enable

Description

- Enables the screen saver.

Syntax

- **ScreenSaver.Enable**

Examples

- ScreenSaver.Enable

Prev Up Next

ScreenSaver.Start

Description

- Starts the screen saver.

Syntax

- **ScreenSaver.Start**

Examples

- ScreenSaver.Start

Prev Up Next

ScreenSaver.Stop

Description

- Stops the screen saver.

Syntax

- **ScreenSaver.Stop**

Examples

- ScreenSaver.Stop

ScreenSaver. TempDisable

Description

- Disables the saver until the mouse is moved.

Syntax

- **ScreenSaver.TempDisable**

Examples

- ScreenSaver.TempDisable

Notes

- The TempDisable command is normally used with a screen corner hotkey. Moving the mouse to the hotkey screen corner which activates the command will disable the saver until the mouse is moved again.

PrevUp Next

Shutdown.Dialog

Description

- Shows the standard "Shut Down Windows" exit dialog.

Syntax

- **Shutdown.Dialog**

Examples

- Shutdown.Dialog

Shutdown.Hibernate

Description

- Puts the computer into hibernate power mode.

Syntax

- **Shutdown.Hibernate**

Examples

- Shutdown.Hibernate

Prev Up Next

Shutdown. LockWorkStation

Description

- Sign off user and lock workstation.

Syntax

- **Shutdown.LockWorkStation**

Examples

- Shutdown.LockWorkStation

PrevUp Next

Shutdown.Logoff

Description

- Logs the current Windows user off.

Syntax

- **Shutdown.Logoff**

Examples

- Shutdown.Logoff

Shutdown.PowerPro

Description

- Exits PowerPro.

Syntax

- **Shutdown.PowerPro**

Examples

- Shutdown.PowerPro

Notes

- If a command list named "PProShutdown" exists, it is run.

Prev Up Next

Shutdown.Reboot

Description

- Shuts down Window and reboots the system.

Syntax

- **Shutdown.Reboot**

Examples

- Shutdown.Reboot

Shutdown.Restart

Description

- Shuts down system with warm Windows restart.

Syntax

- **Shutdown.Restart**

Examples

- Shutdown.Restart

Prev Up Next

Shutdown.Suspend

Description

- Puts the computer into suspend/standby power mode.

Syntax

- **Shutdown.Suspend**

Examples

- Shutdown.Suspend

Prev Up Next

Shutdown.Windows

Description

- Shuts down windows.

Syntax

- **Shutdown.Windows**

Examples

- Shutdown.Windows

Timer.Autosave

Description

- Changes the interval between which timers are autosaved to the pcf file.

Syntax

- **Timer.Autosave(*n*)**

Parameters

n	(integer) new time between saves, in seconds Specify 0 to stop PowerPro autosaving timers.

Examples

- Timer.Autosave(600)
  - Changes the timer autosave interval to 600 seconds.

Notes

- PowerPro normally autosaves running timers to the current .pcf file every 300 seconds.

Timer.Clear

Description

- Zeros (clears) the specified timer or timers.

Syntax

- **Timer.Clear(*timers*)**

Parameters

timers	(string) the timers to clear, as a single string with no spaces between each timer listed

Examples

- Timer.Clear("a")
  - Clears timer a.

- Timer.Clear("fkt")
  - Clears timers f, k & t.

Prev Up Next

Timer.Set

Description

- Starts, stops or toggles the specified timer(s) and sets the value.

Syntax

- **Timer.Set(*"[cAxn][cAddSub]sTimers newHMS"*)**

Parameters

cAxn

(string) single character representing the action to take

Optional, if omitted, the timer state is unchanged.

Possible Values

+	timer(s) in sTimer parameter are started
-	timer(s) in sTimer parameter are stopped
*	timer(s) in sTimer parameter are toggled

cAddSub

(string) single character indicating whether to add, subtract or absolutely set the timer value

Optional, if omitted, the timer is set to an absolute time as specified by newHMS parameter.

Possible Values

@ value	add the value in newHMS parameter to current timer
$	subtract the value in newHMS parameter from current timer value

sTimers	(string) single string of timer ID's to be set, with no spaces between them
newHMS	(string) the new timer value, as 3 space-separated integers: hours, minutes & seconds

Examples

- Timer.Set("+a 0 0 0")
  - Clears timer a and starts it.
- Timer.Set("+a 0 0 120")
  - Starts timer a at 120 seconds.
- Timer.Set("bf 0 10 20")
  - Resets timers b and f to 10 minutes, 20 seconds, leaving their running/stopped state unchanged.
- Timer.Set("-c 1 0 0")
  - Stops timer c and sets its value to one hour.
- Timer.Set("@aq 2 3 0")
  - Adds 2 hours and 3 minutes to timers a and q.
- Timer.Set("$q 2 3 0")
  - Subtracts 2 hours and 3 minutes from timer q.

Prev Up Next

Timer.Start

Description

- Starts the specified timer or timers.

Syntax

- **Timer.Start(*timers*)**

Parameters

timers	(string) the timers to start, as a single string with no spaces between each timer listed

Examples

- Timer.Start("a")
  - Starts timer a.

- Timer.Start("dgk")
  - Starts the three timers d, g & k.

Timer.StartStop

Description

- Toggles the running of the specified timer or timers.

Syntax

- **Timer.StartStop(*timers*)**

Parameters

timers	(string) the timers to stop, as a single string with no spaces between each timer listed

Examples

- Timer.StartStop("a")
  - Starts timer a if it is stopped, else stops it.

- Timer.StartStop("dgk")
  - Starts the three timers d, g & k, if they are stopped, else stops them.

Timer.Stop

Description

- Stops the specified timer or timers.

Syntax

- **Timer.Stop(*timers*)**

Parameters

timers	(string) the timers to stop, as a single string with no spaces between each timer listed

Examples

- Timer.Stop("a")
  - Stops timer a.

- Timer.Stop("dgk")
  - Stops the three timers d, g & k.

TrayIcon.Dump

Description

- Generates a file in the PowerPro folder called "trayicons.txt", containing a list of all tray icons.

Syntax

- **TrayIcon.Dump**

Examples

- TrayIcon.Dump

PrevUp Next

TrayIcon.Hide

Description

- Hides specified tray icon from PowerPro & system tray.

Syntax

- **TrayIcon.Hide(*[^]caption_list[(id)]*)**

Parameters

^	(string) a single carat character, ^, prepended to the icon name to avoid errors if the icon is not found
caption_list	(string) identifies which tray icon window to work with **Possible Values** ❍ The icon's window can be specified via its: *caption* *caption *caption* caption* cap*tion *exe name* =exename *window class* c=windowclass
id	(mixed) Indicates which tray icon to use for a window when which controls more than one tray icon. Optional.

Examples

- TrayIcon.Hide("=taskmgr")
  - ❍ Hides the Task Manager icon from both PowerPro's & Windows' system trays.

Notes

- Commands can still be sent to hidden icons.

TrayIcon.Left

Description

- Simulates a left mouse click on the specified tray icon.

Syntax

- **TrayIcon.Left(*[^]caption_list[(id)]*)**

Parameters

^	(string) a single carat character, ^, prepended to the icon name to avoid errors if the icon is not found
caption_list	(string) identifies which tray icon window to work with **Possible Values** ○ The icon's window can be specified via its: *caption* — *caption, *caption*, caption*, cap*tion *exe name* — =exename *window class* — c=windowclass
id	(mixed) Indicates which tray icon to use for a window when which controls more than one tray icon. Optional.

Examples

- TrayIcon.Left("avast*")
  - Simulates a left-click on the Avast! tray icon.

- TrayIcon.Show("^avast*")
  - Sends a left-click to the Avast! icon, hiding any errors if Avast! is not running.

TrayIcon.LeftDouble

Description

- Simulates a double left mouse click on the specified tray icon.

Syntax

- **TrayIcon.LeftDouble(*[^]caption_list[(id)]*)**

Parameters

^	(string) a single carat character, ^, prepended to the icon name to avoid errors if the icon is not found
caption_list	(string) identifies which tray icon window to work with **Possible Values** ❍ The icon's window can be specified via its: *caption* *caption, *caption*, caption*, cap*tion *exe name* =exename *window class* c=windowclass
id	(mixed) Indicates which tray icon to use for a window when which controls more than one tray icon. Optional.

Examples

- TrayIcon.LeftDouble("avast*")
  - ❍ Simulates a double left-click on the Avast! tray icon.

TrayIcon.Refresh

Description

- Refreshes the tray icons.

Syntax

- **TrayIcon.Refresh**

Examples

- TrayIcon.Refresh

Notes

- Useful if tray icons stop functioning.

Prev Up Next

TrayIcon.Right

Description

- Simulates a right mouse click on the specified tray icon.

Syntax

- **TrayIcon.Right(*[^]caption_list[(id)]*)**

Parameters

^	(string) a single carat character, ^, prepended to the icon name to avoid errors if the icon is not found
caption_list	(string) identifies which tray icon window to work with **Possible Values** ❍ The icon's window can be specified via its: *caption* — *caption, *caption*, caption*, cap*tion *exe name* — =exename *window class* — c=windowclass
id	(mixed) Indicates which tray icon to use for a window when which controls more than one tray icon. Optional.

Examples

- TrayIcon.Right("avast*")
  - ❍ Simulates a right-click on the Avast! tray icon.

- TrayIcon.Right("=explorer(2)")
  - ❍ Simulates a right-click on the icon belonging to the Explorer-type window with id=2.

TrayIcon.RightDouble

Description

- Simulates a double right mouse click on the specified tray icon.

Syntax

- **TrayIcon.RightDouble(*[^]caption_list[(id)]*)**

Parameters

^	(string) a single carat character, ^, prepended to the icon name to avoid errors if the icon is not found
caption_list	(string) identifies which tray icon window to work with **Possible Values** ❍ The icon's window can be specified via its: *caption* *caption *caption* caption* cap*tion *exe name* =exename *window class* c=windowclass
id	(mixed) Indicates which tray icon to use for a window when which controls more than one tray icon. Optional.

Examples

- TrayIcon.RightDouble("avast*")
  - ❍ Simulates a double right-click on the Avast! tray icon.

TrayIcon.Show

Description

- Forces a tray icon to be shown in both PowerPro's & the system's tray.

Syntax

- **TrayIcon.Show(*[^]caption_list[(id)]*)**

Parameters

^

(string) a single carat character, ^, prepended to the icon name to avoid errors if the icon is not found

caption_list

(string) identifies which tray icon window to work with

Possible Values

- The icon's window can be specified via its:

caption	*caption *caption* caption* cap*tion
exe name	=exename
window class	c=windowclass

id

(mixed) Indicates which tray icon to use for a window when which controls more than one tray icon.

Optional.

Examples

- TrayIcon.Show("=taskmgr")
  - Shows the Task Manager icon in both PowerPro's & Windows' system trays.

- TrayIcon.Show("^=taskmgr")
  - Shows the Task Manager icon in both PowerPro's & Windows' system trays, hiding any errors if Task Manager is not running.

Next

Prev Up

Vdesk.Arrange

Description

- Displays a PowerPro Virtual Desktop Arrange window showing all desktops & windows on them.

Syntax

- **Vdesk.Arrange**

Examples

- Vdesk.Arrange

Vdesk.Clear

Description

- Clears the current desktop (closes all windows on it).

Syntax

- **Vdesk.Clear**

Examples

- Vdesk.Clear

Notes

- Vdesk.Clear does not close windows that are visible on all desktops, as set via the PowerPro Config GUI > Desktops > Setup.

PrevUp Next

Vdesk.ClearAllClose

Description

- Move all windows to current desktop and then closes them.

Syntax

- **Vdesk.ClearAllClose**

Examples

- Vdesk.ClearAllClose

Notes

- Vdesk.Clear does not close windows that are visible on all desktops, as set via the PowerPro Config GUI > Desktops > Setup.

PrevUp Next

Vdesk.Consolidate

Description

- Move all windows to current desktop & closes all other desktops.

Syntax

- **Vdesk.Consolidate**

Examples

- Vdesk.Consolidate

PrevUp Next

Vdesk. CreateOrSwitchTo

Description

- Switches to the specified desktop, if it exists, otherwise creates a new desktop named after the specified command list, then runs the commands on the list to populate the desktop.

Syntax

- **Vdesk.CreateOrSwitchTo(*deskname*)**

Parameters

deskname	(string) vdesk name or number

Examples

- Vdesk.CreateOrSwitchTo("work")
  - Switches to the desk named "work" if it exists, otherwise creates it.

See Also

- Vdesk.CreateOrSwitchTo() is the same as Vdesk.NewFromList().

Vdesk.Lock

Description

- Locks all visible windows matching the specified caption list, so that they are visible on all desktops.

Syntax

- **Vdesk.Lock(*cl*)**

Parameters

cl	(string) list of windows to lock

Examples

- Vdesk.Lock("=notepad,*explorer*")
  - Locks all visible Notepad & Explorer windows to all desktops.

PrevUp Next

Vdesk.Menu

Description

- Displays the virtual desktop menu.

Syntax

- **Vdesk.Menu**

Examples

- Vdesk.Menu

Notes

- The virtual desktop menu can also be shown by Shift+right-clicking on a PowerPro bar.

PrevUp Next

Vdesk.MoveActive

Description

- Moves the active window to the specified desktop.

Syntax

- **VDesk.MoveActive(*deskname*)**

Parameters

deskname	(string) desktop name or number The desktop "deskname" must already exist.

Examples

- VDesk.MoveActive("work")
  - Moves the active window to the desktop named "work".

- VDesk.MoveActive(2)
  - Moves the active window to the desktop #2.

Version Compatibility

- Integer deskname parameter introduced in PowerPro 4.1.0.5.

Vdesk.MoveAll

Description

- Moves all windows on the source desktop matching the specified caption list to the destination desktop.

Syntax

- **Vdesk.MoveAll(*from_desk, to_desk, cap_list*)**

Parameters

from_desk	(mixed) source desktop, can be either: ❍ (string) desktop name ❍ (integer) desktop number
to_desk	(mixed) destination desktop, as per from_desk
cap_list	(string) comma-separated string caption

Keywords

* means the current desktop when used with:

- ❍ from_desk
- ❍ to_desk

means all windows, when used with cap_list

Examples

- Vdesk.MoveAll(2, "*", "=notepad,*explorer*")
  - ❍ Moves all windows matching "=notepad" or "*explorer*" on desktop 2, to the current desktop.

Notes

- current prob's
  - ❍ Windows not in cap_list hidden if they're on from_desk, but current desktop is to_desk.

- Minimised windows not moved when "*" keyword for all windows used.

PrevUp Next

Vdesk.MoveAutorun

Description

- Moves last autorun window to the specified desktop.

Syntax

- **VDesk.MoveAutorun(*deskname*)**

Parameters

deskname	(string) desktop name or number The desktop "deskname" must already exist.

Examples

- VDesk.MoveAutorun("work")
  - Moves last autorun window to the desktop named "work".

- VDesk.MoveAutorun(2)
  - Moves last autorun window to the desktop #2.

Version Compatibility

- Integer deskname parameter introduced in PowerPro 4.1.0.5.

Vdesk.New

Description

- Creates a new desktop.

Syntax

- **Vdesk.New(*deskname*)**

Parameters

deskname	(string) name to give the new desktop Optional, if omitted, desktop will just be numbered but not named.

Examples

- Vdesk.New
  - Creates a new desktop, which will be given the next available desktop number.

- Vdesk.New("work")
  - Creates a new desktop named "work".

See Also

- Vdesk.CreateOrSwitchTo()

Vdesk.NewFromList

Description

- Switches to the specified desktop, if it exists, otherwise creates a new desktop named after the specified command list, then runs the commands on the list to populate the desktop.

Syntax

- **Vdesk.NewFromList(*deskname*)**

Parameters

deskname	(string) vdesk name or number

Examples

- Vdesk.NewFromList("work")
  - Creates a new desktop named "work", or switches to it if it already exists.

See Also

- Vdesk.NewFromList() is the same as Vdesk.CreateOrSwitchTo().

Vdesk.Next

Description

- Activates the next virtual desktop.

Syntax

- **Vdesk.Next**

Examples

- Vdesk.Next

Vdesk.Previous

Description

- Activates the previous virtual desktop.

Syntax

- **Vdesk.Previous**

Examples

- Vdesk.Previous

Vdesk.ShowMenu

Description

- Shows a menu of desktops & windows, then either:
  - the selected window is moved to the current desktop.
  - the selected desktop is activated.

Syntax

- **Vdesk.ShowMenu**

Examples

- Vdesk.ShowMenu

Prev Up Next

Vdesk.SwitchMenu

Description

- Shows a menu of desktops and windows, & the selected one is activated.

Syntax

- **Vdesk.SwitchMenu**

Examples

- Vdesk.SwitchMenu

Prev Up Next

Vdesk.SwitchTo

Description

- Switches to the specified desktop.

Syntax

- **Vdesk.SwitchTo(*deskname*)**

Parameters

deskname	(mixed) name or number of the target desktop

Examples

- Vdesk.SwitchTo("mydesk")
  - Switches to the desktop named "mydesk".

- Vdesk.SwitchTo(3)
  - Switches to desktop #3.

Prev Up Next

Vdesk.ReplaceByList

Description

- Clears the current desktop, renames it to the specified command list, then runs the commands on the named list to populate the desktop.

Syntax

- **Vdesk.ReplaceByList(*list_name*)**

Parameters

list_name	(string) name of a command list

Examples

- Vdesk.ReplaceByList("stuff")

Prev Up Next

Vdesk.Unlock

Description

- Unlocks all windows matching the specified caption list, so that they are only visible on one desktop.

Syntax

- **Vdesk.Unlock(*cl*)**

Parameters

cl	(string) list of windows to unlock

Examples

- Vdesk.Unlock("=notepad,*explorer*")
  - Unlocks all Notepad & Explorer windows.

PrevUp Next

Wait.Activity

Description

- Waits for mouse or keyboard activity.

Syntax

- **Wait.Activity**

Examples

- Wait.Activity
  messagebox("ok", "Welcome Back!", "PP Greeting")

Notes

- Always waits at least 3 seconds to ignore activity associated with launching the command.

PrevUp Next

Wait.For

Description

- Wait for specified length of time, or for an expression to evaluate to true.

Syntax

- **Wait.For(*n*, *expr*)**
  - Waits up to n milliseconds for expression expr to become true.

- **Wait.For(*n*)**
  - Wait for n milliseconds.

- **Wait.For(*expr*)**
  - Waits for expression expr to become true.

Parameters

n	(integer) length of time in milliseconds
expr	(bool) expression to evaluate

Examples

- Wait.For(300)
- Wait.For(activewindow("=notepad"))
- Wait.For(300, activewindow("=notepad"))

Notes

- Only one Wait.For() can be running at a time.
- Nested Wait.For()'s are always stopped in reverse order of issuance.
- Wait.For() will not accept an argument from a function call as its n parameter [length of time to wait] when the syntax being used is Wait.For(n), where n is an integer.

  i.e. The following will not wait, no matter what arg(1) evaluates to:

Wait.For(arg(1))

To use the value of arg(1) in the Wait.For() call, use:
- Wait.ForInterval()
- the &() format: Wait.For(&(arg(1))) [deprecated]

See Also

- Wait.ForInterval()
    - in this section

PrevUp Next

Wait.ForInterval

Description

- Waits for specified amount of time.

Syntax

- **Wait.ForInterval(*n*)**

Parameters

n	(integer) time to wait, in milliseconds

Examples

- Wait.ForInterval(300)
  - Waits for 300 milliseconds.

- Wait.ForInterval(arg(2)*4)
  - Waits for the value of "arg(2) times 4" milliseconds.

Notes

- Wait.ForInterval() accepts as a parameter an argument from a function call:
  Wait.ForInterval(arg(1))

  which the normal Wait.For() command doesn't allow.

Wait.Message

Description

- Displays a message box with specified msg and, optionally, a countdown timer starting at n seconds, & waits for user to press a button to determine whether to continue.

Syntax

- **Wait.Message(*n, msg*)**
  - Shows message msg for up to n seconds.

- **Wait.Message(*msg*)**
  - Shows message msg for 1 second.

Parameters

n	(integer) length of countdown, in seconds. If countdown reaches 0, then the wait ends and the next PowerPro command is run. Optional, default is 1.
msg	(string) message to display in message box

Examples

- Wait.Message("Shutting down...")
  Shutdown.Windows

- Wait.Message(10, "Rebooting in 10 seconds...")
  Shutdown.Reboot

Wait.Quit

Description

- Quit all waits.

Syntax

- **Wait.Quit**

Examples

- Wait.Quit

PrevUp Next

Wait.Ready

Description

- Wait for specified program(s) to be ready to accept input.

Syntax

- **Wait.Ready(*cl*)**

Parameters

cl	window id or caption list

Examples

- Wait.Ready("*firefox*")

Notes

- Waits for a maximum of 10 seconds.
- Cannot be used in scripts called from another script.

Prev Up Next

Wait.Until

Description

- Wait until specified length of time, or until an expression evaluates to true.

Syntax

- **Wait.Until(*n, expr*)**
  - Waits until expression expr becomes true, up to a maximum of n milliseconds.

- **Wait.Until(*n*)**
  - Wait until n milliseconds.

- **Wait.Until(*expr*)**
  - Waits until expression expr becomes true.

Parameters

n	(integer) length of time in milliseconds
expr	(string) expression to evaluate

Examples

- Wait.For(300)
- Wait.For(activewindow("=notepad"))
- Wait.For(300, anywindow("=notepad"))

Notes

- Wait.Until() cannot be used inside a do( ) command.
- Multiple independent waits can run concurrently.

Prev Up Next

Wallpaper.Change

Description

- Changes the wallpaper to another file from the same folder as the current desktop wallpaper file.

Syntax

- **Wallpaper.Change(*order*)**

Parameters

order	(string) determines how the next file is selected

Possible Values

*	changes to the next file in Name sequence
omitted	changes to a random file

Examples

- Wallpaper.Change
  - Changes the desktop wallpaper to a random file from the same folder as the current wallpaper.

- Wallpaper.Change("*")
  - Changes the desktop wallpaper to the next file from the same folder as the current wallpaper.

Notes

- IrfanView (and possibly other image management software) has a "Set as Wallpaper" feature, which stores a copy of the selected image in its own folder, overwriting it each time a new image is selected. As such, PowerPro's Wallpaper.Change feature will appear not to work if desktop wallpaper has been set via such programs.

Prev

Up Next

Wallpaper.ChangeTo

Description

- Changes the desktop wallpaper to the specified file, & saves the details in the current .pcf.

Syntax

- **Wallpaper.ChangeTo(*filespec*)**

Parameters

filespec	(string) new wallpaper file

Examples

- Wallpaper.ChangeTo(?"d:\textures\pattern_003.bmp")
  - Changes the wallpaper to "pattern_003.bmp" & stores that file name in the current .pcf.

See Also

- Wallpaper.Show(), which is similar but does not store the new file name.

Prev Up Next

Wallpaper.Show

Description

- Changes the desktop wallpaper to the specified file, but does not save the new file details in the current .pcf.

Syntax

- **Wallpaper.Show(*filespec*)**

Parameters

filespec	(string) new wallpaper file

Examples

- Wallpaper.Show(?"d:\textures\pattern_003.bmp")
  - Changes the wallpaper to "pattern_003.bmp".

See Also

- Wallpaper.ChangeTo(), which is similar but also stores the new file name.

Wallpaper.Style

Description

- Sets the wallpaper file layout

Syntax

- **Wallpaper.Style(*display*)**

Parameters

display	(string) sets picture display style

Possible Values

center	
tile	
stretch	

Examples

- Wallpaper.Style("tile")
  - Sets wallpaper display style to "Tile".

PrevUp Next

Window.AutoMin

Description

- Minimises all windows matching the specified caption list in one of 2 ways depending on whether each individual window is listed in the "Auto tray min" list on the PowerPro Config GUI > Setup tab or not, either minimising the window to the tray, or performing an ordinary minimise.

Syntax

- **Window.Automin(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Automin("active")
  - Minimises the active window to the tray or to the task bar.

Window.Back

Description

- Sends all windows matching the specified caption list to the bottom of the stack of displayed windows.

Syntax

- **Window.Back(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Back("active")
  - Sends the active window to the back.

PrevUp Next

Window.BackShow

Description

- Toggles backmost/foremost setting of each window matching the specified caption list, sending it to the back if foremost, else activating it.

Syntax

- **Window.BackShow(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.BackShow("*notepad")
  - Sends each visible Notepad window to the back of the stack if it's active, else activates it.

Notes

- Each matching window is toggled separately, with respect to its own current state.

Prev Up Next

Window.Center

Description

- Centres the specified window(s) within the full screen area.

Syntax

- **Window.Center(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Center("=pproconf")
  - Centres the PowerPro Config dialog on the screen.

Prev Up Next

Window.Close

Description

- Closes the specified window(s).

Syntax

- **Window.Close(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Close("under")
  - Closes the window under the mouse.

Window.Close2

Description

- Closes the specified window(s) using a different technique than Window.Close().

Syntax

- **Window.Close2(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Close2("under")
  - Closes the window under the mouse.

Notes

- Window.Close2() may work when Window.Close() does not.
- Window.Close2() will also close hidden windows.

Window.CloseForce

Description

- Forces the specified window(s) to close, possibly losing any unsaved information.

Syntax

- **Window.CloseForce(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.CloseForce("*notepad")
  - Forces all Notepad windows to close without saving.

Prev Up Next

Window.Hide

Description

- Makes a window invisible.

Syntax

- **Window.Hide(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Hide("menu")
  - Shows a menu of visible windows, then hides the selected window.

Window.HideShow

Description

- Toggles a window's visibility.

Syntax

- **Window.Hide(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Hide("menu")
  - Shows a menu of visible windows, then hides the selected window.

Notes

- Take care when applying Window.HideShow() to multiple windows, as there are many windows which should normally remain invisible.

Window.Max

Description

- Maximises the specified window(s).

Syntax

- **Window.Max(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Max("under")
  - Maximises the window under the mouse.

- Window.Max("*metapad")
  - Maximises all visible Metapad windows.

Prev Up Next

Window.MaxNormal

Description

- Toggles the maximised state of the specified window(s).

Syntax

- **Window.MaxNormal(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.MaxNormal("active")
  - Maximises the active window if it is currently unmaximised, else restores it to normal state.

Notes

- Each matching window is toggled separately, with respect to its own current state.

Window.Min

Description

- Minimises the specified window(s).

Syntax

- **Window.Min(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Min("*firefox")
  - Minimises all visible Firefox windows.

Window.MinMemory

Description

- Sets the memory working set for the specified window(s).
- Windows NT only.

Syntax

- **Window.MinMemory(*cl*, *min*, *max*)**

Parameters

cl	(mixed) caption list identifying target window(s)
min	(float) minimum working set size in bytes Optional, see Notes below for more.
max	(float) maximum working set size in bytes Optional, see Notes below for more.

Examples

- Window.MinMemory("active")

Notes

- If the parameters min & max are omitted, or if both are specified as -1, the function temporarily trims the working set of the specified process to zero, essentially swapping the process out of physical RAM memory.
- The virtual memory manager attempts to keep at least the minimum working set size resident in the process whenever the process is active and to keep no more than the maximum memory resident in the process whenever the process is active and memory is in short supply.

Window.MinRestore

Description

- Toggles the minimised state of each window matching the specified caption list.

Syntax

- **Window.MinRestore(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.MinRestore("*metapad")
  - For each visible Metapad window, restores it if minimised, else minimises it.

Notes

- Each matching window is toggled separately, with respect to its own current state.

Prev Up Next

Window.Move

Description

- Sets the specified window(s) to move with the mouse until any mouse button is pressed.

Syntax

- **Window.Move(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Move("under")
  - Sets the window under the mouse to be moved by the mouse's movements until a mouse button is pressed.

Prev Up Next

Window.Normal

Description

- Restores all windows matching the specified caption list to "Normal" size (not minimised or maximised).

Syntax

- **Window.Normal(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Normal("*firefox")
  - Restores all visible Firefox windows.

PrevUp Next

Window.NotTop

Description

- Removes any "Always On Top" settings for all windows matching the specified caption list.

Syntax

- **Window.NotTop(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.NotTop("under")
  - Removes the "Always on top" setting of the window under the mouse.

Prev Up Next

Window.OnTop

Description

- Sets all windows matching the specified caption list to be "Always on top".

Syntax

- **Window.OnTop(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.OnTop("under")
  - Sets the window under the mouse to be on top, so that it is always above other windows.

Prev Up Next

Window.PostMessage

Description

- Sends a WinAPI PostMessage() call to each window matching the specified caption list.

Syntax

- **Window.PostMessage(*msg*, *wp*, *lp*, *han*)**

Parameters

msg	(mixed) the message to be posted

Possible Values

wm_command	
wm_app	
wm_user	
wm_user+n	where n is a number

wp	(integer) specifies additional message-specific information
lp	(integer) specifies additional message-specific information
han	(mixed) handle or class of the window(s) to act on

Return Value

- (boolean)

1	Function succeeded.
0	function failed

Examples

- Window.PostMessage_("wm_command", 1193, 0, "c=i_view32")
  - Toggles the menubar in IrfanView.

- Window.PostMessage(0x112, 0xF170, 2, 0xFFFF)
  - Turns off the screen.
  - Use -1 as the 3rd parameter to turn the screen on, or 1 to make it dimmed.

Notes

- The WinAPI PostMessage() function posts a message in the message queue associated with the thread that created the target window, then returns immediately, without waiting for the message to be processed.
- Precede parameter integer values with "0x" to indicate a hexadecimal number, for example:
  - 0x1f0a
- Note that Window.PostMessage() is generally safer to use than Window.SendMessage(), unless a return is needed from the message handler.

See Also

- Window.SendMessage()
  - *PPSR > Built-in Commands > Commands > Window > Actions > SendMessage*
- **http://msdn.microsoft.com/library/en-us/...andmessagequeuesfunctions/postmessage.asp**

Prev Up Next

Window.Position

Definition

- Repositions and/or resizes target window according to values specified.
- New size & position values may be keywords, a positive or negative number (in quotes), or the equals sign.

Syntax

- **Window.Position(*x, y, w, h, cl*)**

Parameters

x	(string) x-coordinate of window
y	(string) y-coordinate of window
w	(string) width of window
h	(string) height of window
cl	(mixed) caption list identifying target window(s)

Keywords for x, y, w, h

"="	keep current value
"+"	prepended to the value
value is calculated relative to current position	
"center"	window is centred on screen
"hmax"	maximum height
"wmax"	maximum width

Examples

- Window.Position("=", "=", "300", "200", "active")
  Window.Position("= = 300 200", "active")

```
Window.Position("= = 300 200 active")
Window.Position("=", "= 300", "200 active")
```

- Resizes active window to 300x200.
- Note that each of the Window.Position() calls here has the same effect.

  PowerPro accepts any combination of the 5 parameters as separate strings in the Window.Position() call:

  - 5 separate string values
  - a single string containing 5 space-separated values
  - 2, 3, or 4 comma-separated strings, each containing a number of space-separated values

- Window.Position("0", "0", "=", "hmax/2", "=notepad")
  - Moves notepad to top left corner of screen & resizes it to half maximum height.

- Window.Position("+-50", "+-80", "=", "=", "active")
  - Moves active window 50px left and 80px up.

- Window.Position("++100", "++150", "=", "=", "active")
  - Moves active window 100px right and 150px down.

- Window.Position("++100 ++150 = = active")
  - This has the same effect as the above example, but it is written differently, with all parameters given in a single space-separated string.

Notes

- See the Win plugin for other ways to resize & move windows.

Window.Rollup

Description

- Toggles a window's "rolled-up" state, rolling it up to just its caption if it is fully visible, or showing it again if it is already rolled-up.

Syntax

- **Window.Rollup(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Rollup("under")
  - Rolls up the window under the mouse.

Prev Up Next

Window.SendMessage

Description

- Sends a WinAPI SendMessage() call to each window matching the specified caption list.

Syntax

- **Window.SendMessage(*msg*, *wp*, *lp*, *han*)**

Parameters

msg	(mixed) the message to be sent

Possible Values

wm_command	
wm_app	
wm_user	
wm_user+n	where n is a number

wp	(integer) specifies additional message-specific information
lp	(integer) specifies additional message-specific information
han	(mixed) handle or class of the window(s) to act on

Return Value

- (mixed) depends on the message sent

Examples

- Window.SendMessage("wm_command", 40046, 0, "c=winamp v1.x")
  - Pauses Winamp.

- Window.SendMessage("wm_user", 1, 105, "c=winamp v1.x")
  - Sets variable SendMessage to length of current track

Notes

- The WinAPI function, SendMessage, sends the specified message to the target window(s), calling the window procedure & waiting for it to process the message before returning.
- Precede parameter integer values with "0x" to indicate a hexadecimal number, for example:
  - 0x1f0a
- Note that Window.PostMessage() is generally safer to use than Window.SendMessage(), unless a return is needed from the message handler.

See Also

- Window.PostMessage()
  - *PPSR > Built-in Commands > Commands > Window > Actions > PostMessage*
- **http://msdn.microsoft.com/library/en-us/...andMessageQueuesFunctions/SendMessage.asp**

Prev Up Next

Window.SetPriority

Description

- Sets process priority of specified window.

Syntax

- **Window.SetPriority(*amount*, *cl*)**

Parameters

amount	(string) new process priority

Possible Values

idle	lowest
below	
normal	
above	
high	highest

cl	(mixed) caption list identifying target window(s)

Examples

- Window.SetPriority("high", "=myprog")
  - Sets the process priority of the program MyProg to high.

Window.Show

Description

- Activates the specified window & shows it if hidden.

Syntax

- **Window.Show(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Show("=notepad")
  - Shows the first notepad found.

- Window.Show("menu =notepad")
  - Shows a menu of Notepad windows, & one selected from the list is brought to the front.

Notes

- PowerPro accepts anything for the 2nd part of a Window.Show("menu...") statement, including nothing. If it doesn't recognise the word or reference, an empty menu will be shown.

Prev Up Next

Window.Size

Description

- Enables the user to size the window by moving the mouse; click any mouse button to stop.

Syntax

- **Window.Size(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.Size("*metapad")
  - Allows the user to resize a Metapad window by moving the mouse.

Prev Up Next

Window.TopNotTop

Description

- Toggles the "Always on top" setting for each window matching the specified caption list.

Syntax

- **Window.TopNotTop(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.OnTop("under")
  - Sets the window under the mouse to be on top, so that it is always above other windows.

Notes

- Each matching window is toggled separately, with respect to its own current state.

Window.Trans

Description

- Makes all windows matching the specified caption list transparent.
- Windows 2000 and XP only.

Syntax

- **Window.Trans(*n*, *cl*)**

Parameters

n	(number) transparency value -254<n<255 where: *0* — no transparency *255* — completely transparent *negative values* — toggle transparency setting
cl	(mixed) caption list identifying target window(s)

Examples

- Window.Trans(-254, "HTML Help*PowerPro")
  - Toggles the transparency setting of the PowerPro Help CHM file between 0 (no transparency) and 254 (transparent).

See Also

- Window.TransMouse()
  - *PPSR > Built-in Commands > Commands > Window > Actions > TransMouse*

PrevUp Next

Window.TransMouse

Description

- Makes all windows matching the specified caption list transparent, as well as making all mouse clicks pass through them.
- Windows 2000 and XP only.

Syntax

- **Window.TransMouse(*n*, *cl*)**

Parameters

n	(number) transparency value -254<n<255 where *0* — no transparency *255* — completely transparent *negative values* — toggle transparency setting
cl	(mixed) caption list identifying target window(s)

Examples

- Window.TransMouse(-80, "HTML Help*PowerPro")
  - Toggles the transparency setting of the PowerPro Help CHM file between 0 and 100, as well as making mouse clicks pass through it to the window below.

See Also

- Window.Trans()
  - *PPSR > Built-in Commands > Commands > Window > Actions > Trans*

PrevUp Next

Window.TrayMin

Description

- Minimizes all windows matching a caption list to the system tray.

Syntax

- **Window.TrayMin(*cl*)**

Parameters

cl	(mixed) caption list identifying target window(s)

Examples

- Window.TrayMin("*firefox")
  - Tray-minimises all firefox windows.
