

# Gill IT Utopia Power Manager

## Scripting User Guide

Copyright © 1998-2011



## Contents

Gill IT Utopia Power Manager .....	1
Scripting User Guide .....	1
Script reference .....	4
Function: Get-ProcessWindowHandle .....	4
Function: Get-WindowHandle .....	4
Function: Get-WindowHandleWait .....	4
Function: Get-WindowHandlesForClass .....	5
Function: Get-ActiveControl .....	5
Function: Get-Control .....	5
Function: Get-ControlRecursive .....	5
Function: Get-ControlText.....	6
Function: Send-Close .....	6
Function: Send-SysClose.....	6
Function: Send-CloseWindowWithEscape .....	7
Function: Send-CloseChildWindowsByHandle .....	7
Function: Send-CloseChildWindowsByClassName .....	7
Function: Send-CloseProcessChildWindows .....	8
Function: Get-WaitWindowClose.....	8
Function: Send-Escape .....	8
Function: Send-EscapeToChildControls.....	8
Function: Send-EscapeToProcess .....	9
Function: Send-Click .....	9
Function: Send-Command.....	10
Function: Send-SysCommand .....	10
Function: Send-Menu .....	10
Function: Set-ActiveWindow .....	11
Function: Set-ForegroundWindow.....	11
Function: Send-Chars .....	12
Function: Send-ControlText.....	12
Function: Send-Enter .....	13
Function: Remove-ProcessByHandle.....	13
Function: Remove-ProcessById .....	13

Function: Remove-HiddenProcess.....	13
Function: Get-ReadyToSave.....	13
Auto Window Processor .....	15
Function: Add-AutoClickWindow.....	15
Function: Add-AutoCloseWindow.....	16
Function: Start-ProcessWindows.....	16
Function: Reset-AutoWindowProcessor.....	16
WPF Applications /Ribbon .....	17
Function: Get-AutomationElement.....	17
Function: Get-AutomationElementHandle.....	17
Function: Get-AutomationElementProcessId.....	17
Function: Get-ElementByName.....	17
Function: Get-ElementById.....	18
Function: Get-ElementByClass.....	18
Function: Set-AutomationElementText.....	18
Function: Execute-ElementByName.....	18
Function: Execute-ElementById.....	19
Function: Execute-ElementByClass.....	19
Utility Functions .....	20
Function: Get-WindowVersion.....	20

## Script reference

### Function: Get-ProcessWindowHandle

Retrieve the window handle for a process

#### Syntax

```
$windowHandle = Get-ProcessWindowHandle <ProcessName>
```

#### Parameters

ProcessName : Name of the process

#### Example

```
$windowHandle = Get-ProcessWindowHandle "notepad"
```

### Function: Get-WindowHandle

Returns the window handle for the matching window name (the window's title).  
Returns 0 if match is not found.

#### Syntax

```
$SaveAsHandle = Get-WindowHandle <WindowName>
```

#### Parameters

WindowName : String to specify the window name (the window's title).

#### Example

```
$SaveAsHandle = Get-WindowHandle "Save As"
```

### Function: Get-WindowHandleWait

Returns the window handle for the matching window name (the window's title).  
If a match is not found the function will keep searching until the timeout period elapses.  
Returns 0 if match is not found.

#### Syntax

```
$SaveAsHandle = Get-WindowHandleWait <WindowName> <Timeout>
```

#### Parameters

WindowName : String to specify the window name (the window's title)

Timeout: Time in seconds before the function returns 0

#### Example

```
$SaveAsHandle = Get-WindowHandleWait "Save As" 5
```

## Function: Get-WindowHandlesForClass

Returns a list of window handle for the matching a class name.

### Syntax

```
$messageHandles = Get-WindowHandlesForClass <Class name>
```

### Parameters

Class name : String to specify the window class name.

### Example

```
$messageHandles = Get-WindowHandlesForClass "OpusApp"
```

## Function: Get-ActiveControl

Returns handle of the active control in a window.

### Syntax

```
$activeHandle = Get-ActiveControl <WindowHandle>
```

### Parameters

WindowHandle : String to specify the window handle.

### Example

```
$Filename = Get-ActiveControl $SaveAsHandle
```

## Function: Get-Control

Returns handle of control within a window which matches its text

### Syntax

```
$ControlHandle = Get-Control <WindowHandle> <Text>
```

### Parameters

WindowHandle : String to specify the window handle.

Text: String to match the control

### Example

```
$SaveButton = Get-Control $ConfirmHandle "&Yes"
```

## Function: Get-ControlRecursive

Returns handle of control within a window which matches its text.  
Searches child controls for a match.

### Syntax

```
$ControlHandle = Get-ControlRecursive <WindowHandle> <Text>
```

### Parameters

WindowHandle : String to specify the window handle.

Text: String to match the control

### Example

```
$YesButton = Get-ControlRecursive $ConfirmSaveAsHandle "&Yes"
```

## Function: Get-ControlText

Returns the control text/title.

### Syntax

```
$Text = Get-ControlText <WindowHandle>
```

### Parameters

WindowHandle : String to specify the window handle.

### Example

```
$Text = Get-ControlText $SaveAsHandle
```

## Function: Send-Close

Send (WM\_COMMAND) Close to a window handle.

### Syntax

```
Send-Close <WindowHandle>
```

### Parameters

WindowHandle : String to specify the window handle.

### Example

```
Send-Close $windowHandle
```

## Function: Send-SysClose

Send (WM\_SYSCOMMAND) Close to a window handle.

### Syntax

```
Send-SysClose <WindowHandle>
```

### Parameters

WindowHandle : String to specify the window handle.

### Example

Send-SysClose \$windowHandle

### Function: Send-CloseWindowWithEscape

Send "Escape" to a window handle. If this request fails to close the window then "Escape" is sent to all the child controls.

#### Syntax

Send-CloseWindowWithEscape <WindowHandle>

#### Parameters

WindowHandle : String to specify the window handle.

#### Example

Send-CloseWindowWithEscape \$windowHandle

### Function: Send-CloseChildWindowsByHandle

Closes all the child windows/dialogs which belong to a window handle.

#### Syntax

Send-CloseChildWindowsByHandle <WindowHandle>

#### Parameters

WindowHandle : String to specify the window handle. Only the child windows will be closed.

#### Example

Send-CloseChildWindowsByHandle \$windowHandle

### Function: Send-CloseChildWindowsByClassName

Closes all the child windows/dialogs which match the given class name.

#### Syntax

Send-CloseChildWindowsByClassName <Class name>

#### Parameters

Class Name : String to specify the class name. Only the child windows will be closed.

#### Example

Send-CloseChildWindowsByClassName "rctrl\_renwnd32"

## Function: Send-CloseProcessChildWindows

Closes all the child windows/dialogs which match the given process name.

### Syntax

```
Send-CloseProcessChildWindows <ProcessName>
```

### Parameters

ProcessName : Name of the process. Only the child windows will be closed.

### Example

```
Send-CloseProcessChildWindows "Excel"
```

## Function: Get-WaitWindowClose

Waits for a window to close given a window handle.

### Syntax

```
Get-WaitWindowClose <Handle> <Timeout>
```

### Parameters

Handle : String to specify the window handle to wait to close

Timeout: Period of time in seconds to wait for.

### Example

```
Get-WaitWindowClose $windowHandle 10
```

## Function: Send-Escape

Send the "Escape" key to the given window handle.

### Syntax

```
Send-Escape <Handle>
```

### Parameters

Handle : String to specify the window handle to receive "Escape"

### Example

```
Send-Escape $windowHandle
```

## Function: Send-EscapeToChildControls

Press "Escape" key for all child controls for the given window handle.

### Syntax



Send-EscapeToChildControls <WindowHandle>

### Parameters

WindowHandle : String to specify the window handle. Only the child controls will be sent the "Escape" key. WindowHandle will not be sent the "Escape" key.

### Example

Send-EscapeToChildControls \$windowHandle

### Function: Send-EscapeToProcess

Call Send-EscapeToChildControls for each matching process name.

### Syntax

Send-EscapeToProcess <ProcessName>

### Parameters

ProcessName : Name of the process.

### Example

Send-EscapeToProcess "Excel"

### Function: Send-Click

Click the control matching the window handle.

### Syntax

Send-Click <Handle>

### Parameters

Handle : String to specify the control handle to receive a mouse click.

### Example

```
$SaveAsHandle = Get-WindowHandleWait "Save As" 5
```

```
$Filename = Get-ActiveControl $SaveAsHandle
```

```
$SaveButton = Get-Control $SaveAsHandle "&Save"
```

```
Set-ControlText $Filename "info.txt"
```

```
Send-Click $SaveButton
```

## Function: Send-Command

Send a windows command to a window/control given its handle.

### Syntax

```
Send-Command <Handle> <wParam> <iParam>
```

### Parameters

Handle : String to specify the handle to receive the command.

wParam: Specifies additional message-specific information

iParam: Specifies additional message-specific information

### Example

The following example will send a Save As command to Notepad

```
$windowHandle = Get-ProcessWindowHandle "Notepad"  
Send-Command $windowHandle 0x00000004 0x00000000
```

## Function: Send-SysCommand

Send a windows SYSCOMMAND to a window/control given its handle.

### Syntax

```
Send-SysCommand <Handle> <wParam> <iParam>
```

### Parameters

Handle : String to specify the handle to receive the command.

wParam: Specifies additional message-specific information

iParam: Specifies additional message-specific information

### Example

The following example will send a SC\_CLOSE command to Notepad

```
$windowHandle = Get-ProcessWindowHandle "Notepad"  
Send-SysCommand $windowHandle 0xF060 0x0
```

## Function: Send-Menu

Select a menu matching the window handle and menu offsets.

### Syntax

```
Send-Menu <Handle> <MainMenuIndex> <SubMenuIndex>
```

### Parameters

Handle : String to specify the window handle to receive a menu click.

MainMenuIndex: Number to denote the main menu. First menu start at 0.

SubMenuIndex: Number to denote the sub menu item to click. First menu item starts at 0.

## Example

The following example will press the following menu option for Notepad: File -> Save As

```
$windowHandle = Get-ProcessWindowHandle "notepad"
Send-Menu $windowHandle 0 3
```

## Function: Set-ActiveWindow

Activate a window matching the window handle.

### Syntax

```
Set-ActiveWindow <Handle>
```

### Parameters

Handle : String to specify the window handle to made active

## Example

The following examples will close Outlook XP mails windows:

```
$dialogHandles = Get-WindowHandlesForClass "bosa_sdm_Microsoft Office Word
11.0"
foreach ($dialog in $dialogHandles)
{
    Set-ActiveWindow $dialog
    Send-Close $dialog
    [System.Threading.Thread]::Sleep(500)
    Set-ActiveWindow $dialog
}
```

## Function: Set-ForegroundWindow

Brings a window to the foreground.

### Syntax

```
Set-ForegroundWindow <Handle>
```

## Parameters

Handle : String to specify the window handle to made active

## Note:

This function has no effect if Windows is locked.

## Function: Send-Chars

Sends a string to a control one character at a time.

## Syntax

```
Send-Chars <Handle> <Message>
```

## Parameters

Handle : String to specify the window handle to receive the char message

Message: string message to send

## Example

The following example will set the file name in a Save As dialog:

```
$SaveAsHandle = Get-WindowHandleWait "Save As" 5  
$Filename = Get-ActiveControl $SaveAsHandle  
Send-Chars $Filename "info.txt"
```

## Function: Send-ControlText

Sends a string to a control.

## Syntax

```
Send-ControlText <Handle> <Message>
```

## Parameters

Handle : String to specify the window handle to receive the char message

Message: string message to send

## Example

The following example will set the file name in a Save As dialog:

```
$SaveAsHandle = Get-WindowHandleWait "Save As" 5  
$Filename = Get-ActiveControl $SaveAsHandle
```

```
Set-ControlText $Filename "info.txt"
```

### Function: Send-Enter

Sends the "Enter" key to a control.

#### Syntax

```
Send-Enter <Handle>
```

#### Parameters

Handle : String to specify the window handle to receive the "Enter" key

### Function: Remove-ProcessByHandle

Kills a process given its handle.

#### Syntax

```
Remove-ProcessByHandle <Handle>
```

#### Parameters

Handle : String to specify the window handle to remove

### Function: Remove-ProcessById

Kills a process given its processId.

#### Syntax

```
Remove-ProcessById <ProcessId>
```

#### Parameters

ProcessId: Process Id to remove.

### Function: Remove-HiddenProcess

Kills hidden process given a process name.

#### Syntax

```
Remove-HiddenProcess <ProcessName>
```

#### Parameters

ProcessName: Process name to remove if hidden

#### Example

```
Remove-HiddenProcess "Excel"
```

### Function: Get-ReadyToSave

This function will:

1. Close all hidden instances
2. Close all child windows
3. Send escape to main window

## Syntax

`Get-ReadyToSave <ProcessName>`

## Parameters

**ProcessName:** Name of process - All instances of this will be processed

## Example

`Get-ReadyToSave "Excel"`

## Auto Window Processor

The Auto Window Processor will iterate through all the windows, clicking buttons and closing windows as specified.

To setup the Auto Window Processor the following functions need to be called:

1. Add-AutoClickWindow – for each button which should be pressed automatically
2. Add-AutoCloseWindow – for each window which should be closed automatically
3. Start-ProcessWindows – Iterates through all the windows once, pressing buttons and closing windows
4. Reset-AutoWindowProcessor – clears all the AutoClickWindows and AutoCloseWindows

The Auto Window Processor supports Regular Expression matching.

### Function: Add-AutoClickWindow

Set the button which should be pressed automatically for a given window/class.

#### Syntax

```
Add-AutoClickWindow <WindowTitle> <ClassName> <Message> <ClickButton>
```

#### Parameters

WindowTitle: Regular Expressions to match Title of the window

ClassName: Regular Expressions to match the Class Name of the window

Message: Regular Expressions to match child Text message of at least one of the child controls in the window

ClickButton: Button to press if conditions are satisfied

#### Example

Close all Open dialogs by pressing the Cancel button:

```
Add-AutoClickWindow "Open" "" "" "Cancel"
```

```
Start-ProcessWindows
```

Press "Close all tabs" button for all IE 8 close dialogs:

```
Add-AutoClickWindow "" "#32770" "Do you want to close all tabs" "Close all &tabs"
```

```
Start-ProcessWindows
```

Press "No" button for Outlook 2007 delete confirmation prompt dialogs:

```
Add-AutoClickWindow "Microsoft Office Outlook" "" "Are you sure you want to permanently delete" "&No"
```

```
Start-ProcessWindows
```

## Function: Add-AutoCloseWindow

Close a window by pressing the Escape key for a given window/class.

### Syntax

```
Add-AutoCloseWindow <WindowTitle> <ClassName> <Message>
```

### Parameters

WindowTitle: Regular Expressions to match Title of the window

ClassName: Regular Expressions to match the Class Name of the window

Message: Regular Expressions to match child Text message of at least one of the child controls in the window

### Example

Close all dialogs:

```
Add-AutoCloseWindow "" "#32770" ""
```

```
Start-ProcessWindows
```

## Function: Start-ProcessWindows

Iterates through all the windows once, pressing buttons and closing windows

### Syntax

```
Start-ProcessWindows
```

### Example

```
Start-ProcessWindows
```

## Function: Reset-AutoWindowProcessor

Clears all the AutoClickWindows and AutoCloseWindows

### Syntax

```
Reset-AutoWindowProcessor
```

### Example

```
Reset-AutoWindowProcessor
```



## WPF Applications /Ribbon

WPF Applications or applications with a Ribbon can not be automated successfully using the above commands. The following functions will enable these to be automated.

### Function: Get-AutomationElement

Returns an automation element for the given handle

#### Syntax

```
Get-AutomationElement <handle>
```

#### Example

```
$windowHandle = Get-ProcessWindowHandle "winword"  
$window = Get-AutomationElement $windowHandle
```

### Function: Get-AutomationElementHandle

Returns the handle for the given automation element

#### Syntax

```
Get-AutomationElementHandle <automation element>
```

#### Example

```
$windowHandle = Get-AutomationElementHandle $window
```

### Function: Get-AutomationElementProcessId

Returns the process Id for the given automation element

#### Syntax

```
Get-AutomationElementProcessId <automation element>
```

#### Example

```
$processId = Get-AutomationElementProcessId $window
```

### Function: Get-ElementByName

Returns the child automation element matching the given name

#### Syntax

```
Get-ElementByName <automation element> <name>
```

#### Example

```
$windowHandle = Get-ProcessWindowHandle "winword"  
$window = Get-AutomationElement $windowHandle
```

```
$saveAsDialog = Get-ElementByName $window "Save As"
```

## Function: Get-ElementById

Returns the child automation element matching the given Id

### Syntax

```
Get-ElementById <automation element> <ID>
```

### Example

```
$windowHandle = Get-ProcessWindowHandle "winword"  
$window = Get-AutomationElement $windowHandle  
$saveAsDialog = Get-ElementByName $window "Save As"  
$filename = Get-ElementById $saveAsDialog "1001"
```

## Function: Get-ElementByClass

Returns the child automation element matching the given class name

### Syntax

```
Get-ElementByClass <automation element> <class name>
```

### Example

```
$windowHandle = Get-ProcessWindowHandle "winword"  
$window = Get-AutomationElement $windowHandle  
$saveAsDialog = Get-ElementByClass $window "#32770"
```

## Function: Set-AutomationElementText

Sets the text for the given element

### Syntax

```
Set-AutomationElementText <automation element> <string>
```

### Example

The following will set the filename name to "example.doc" in the Save As dialog form in Word.

```
$windowHandle = Get-ProcessWindowHandle "winword"  
$window = Get-AutomationElement $windowHandle  
$saveAsDialog = Get-ElementByName $window "Save As"  
$filename = Get-ElementById $saveAsDialog "1001"  
Set-AutomationElementText $filename "example.doc"
```

## Function: Execute-ElementByName

Executes the automation element matching the given name

### Syntax

```
Execute-ElementByName <automation element> <name>
```

### Example

Press Save in MS Word:

```
$windowHandle = Get-ProcessWindowHandle "winword"  
$window = Get-AutomationElement $windowHandle  
Execute-ElementByName $window "Save"  
Execute-ElementByName $window "Close"
```

### Function: Execute-ElementById

Executes the automation element matching the given Id

#### Syntax

```
Execute-ElementById <automation element> <ID>
```

### Example

### Function: Execute-ElementByClass

Executes the automation element matching the given Id

#### Syntax

```
Execute-ElementById <automation element> <Class>
```

### Example

## Utility Functions

### Function: `Get-WindowVersion`

Returns a string representing the version of Windows

#### Syntax

```
Get-WindowVersion
```

Return values include:

- Windows 2000
- Windows XP
- Windows 2003
- Windows Vista
- Windows 7

#### Example

```
$version = Get-WindowVersion  
Write-Host $version
```