

# **MouseviatorHelper.dll**

## Reference Manual

someone may find this helpful

# Table of Contents

1. What the heck is this?.....	2
2. Ok, how do I use it? What if the day is bad one?.....	3
3. Constants exported:.....	4
4. Functions for process manipulation:.....	5
process.closehandle.....	5
process.getaffinity.....	5
process.gethandle.....	6
process.getmemoryinfo.....	7
process.getpriority.....	7
process.getprivatebytes.....	8
process.getid.....	8
process.getids.....	9
process.setaffinity.....	9
process.setpriority.....	10
5. Functions for window manipulation.....	11
window.find.....	11
window.getclientsize.....	12
window.gethandles.....	12
window.getsize.....	13
window.getstyle.....	13
window.gettext.....	13
window.isforeground.....	14
window.isfullscreen.....	14
window.maximize.....	14
window.minimize.....	15
window.move.....	15
window.restore.....	15
window.sendmessage.....	16
window.setforeground.....	16
window.setfullscreen.....	16
window.sewindowed.....	17
window.setposition.....	17
window.setstyle.....	18

# 1. What the heck is this?

This document is supposed to be a reference manual for **MouseviatorHelper.dll** library, which exports some constants and functions that can be used within Lua scripts. The primary goal of the library was to bring functionality required for my EZCA restart script for flight simulator that was not possible to achieve with Lua.

The library currently contains some functions for process manipulation and for window (the actual Windows windows you see on your computer) manipulation. Most of the functions are simply wrappers for win32 functions.

Reference manual? That sounds boring, isn't it? Well, it probably is. Maybe you can use it to make yourself fall asleep. But I hope you may find this collection of characters useful also in it's primary goal, providing you with information of what the functions located in the library do and how to use them.

## 2. Ok, how do I use it? What if the day is bad one?

Most of the functions the library contains are simple wrappers for win32 functions. As mentioned above, there are two groups of functions for now. One for process manipulation (grouped within **process** table) and another for window manipulation (grouped within **window** table). I believe an example will be the best:

```
-- call function for process manipulation like this:
pid = process.getid()
-- call function for window manipulation like this
whandles = window.find("Notepad")
```

The function are designed so they either return the value you expect they should return (and they may return nothing), or they generate an exception in the case of any issues. To tell if there was an exception, use standard Lua pcall construction:

```
print("Trying to find windows by name..")
status, whandles = pcall(window.find,"Notepad")
if status then
  for key, value in pairs(whandles) do
    print(key.." -> "..value)
  end
else
  -- exception occurred, whandles will contain no results, but the error message
  print("There was an error: "..whandles)
end
```

That's it. Not so bad, is it?

### 3. Constants exported:

MouseviatorHelper.dll information. The following constants provides some info about the current library build:

*MVH\_VERSION* - version of the library as text, for example: "1.0b"  
*MVH\_AUTHOR* - author of the library. My name for now :)  
*MVH\_BD* - date when the library was build.  
*MVH\_BT* - time at which the library was build.

Constants for settings process priority. I will not describe them since I think they are self explanatory. They correspond to priorities you can set with task manager for example. Those are used by *process.getpriority* and *process.setpriority* functions.

*P\_ABOVE\_NORMAL*  
*P\_BELOW\_NORMAL*  
*P\_HIGH*  
*P\_IDLE*  
*P\_NORMAL*  
*P\_REALTIME*

Constants for *window.setstyle* function. These tells mentioned function what to do with window style, which is one the function arguments. See the function description for more details.

*WSOP\_ADD* - add style passed to function to current window style.  
*WSOP\_REMOVE* - remove style passed to function from current window style (So we suppose the window has the style and we want to remove it)  
*WSOP\_REPLACE* - replace window style by style passed to function.

Constants for *window.gethandles* and *window.find* functions. These control whether the functions will return handles for all matching windows or only for the matching top level window (windows that are not child of any other window – usually the main program windows).

*WE\_ALL* - will match all windows.  
*WE\_TOPLEVELONLY* - match only top level windows.

## 4. Functions for process manipulation:

All functions for process manipulation are exported to table: **process**. Which means you have to call them like: **process.function**.

### process.closehandle

Usage:

```
-- handle was previously opened by process.gethandle  
process.closehandle (processHandle)
```

Description:

This function will close handle opened by *process.gethandle*.

Parameters:

**processHandle** – a handle to process opened by *process.gethandle*.

Returns:

Nothing if there were no errors. Otherwise, throws exception.

Throws:

Will throw exception if something goes wrong.

### process.getaffinity

Usage:

```
-- handle was previously opened by process.gethandle  
processAffinity, systemAffinity = process.getaffinity (processHandle)
```

Description:

This function will retrieve affinity for process specified by its handle. Use *process.gethandle* to get handle to a process.

Parameters:

**processHandle** – handle of process we want to get affinity for. This has to be greater than 0.

Returns:

**processAffinity** – affinity mask for the specified process.  
**systemAffinity** – affinity mask for the system.

Throws:

Will throw exception if something goes wrong.

Intentionally Left Blank

## process.gethandle

### Usage:

```
-- get handle for process with name "notepad"
processHandle = process.gethandle ("notepad")
-- get handle for process with pid=5020
processHandle = process.gethandle (5020)
-- get handle with specific privileges
local PROCESS_QUERY_INFORMATION = 0x0400
processHandle = process.gethandle (5020, PROCESS_QUERY_INFORMATION)
```

### Description:

This function will return process handle for first process matching given name or process id. The function opens the handle (it has to actually get the handle), so after you do not need it, you have to close it using *process.closehandle*.

### Parameters:

**processName** – name of the process we want to get process id for. Can be just part of name.

**processID** – id of process we want to get handle for.

**dwDesiredAccess** – access rights we want to have to process. See:

[http://msdn.microsoft.com/en-us/library/windows/desktop/ms684880\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684880(v=vs.85).aspx)

for possible values.

### Returns:

Handle of process matching given name or with given id.

### Throws:

Will throw exception if something goes wrong.

Intentionally Left Blank

## process.getmemoryinfo

### Usage:

```
-- handle was previously opened by process.gethandle
memoryinfo = process.getmemoryinfo (processHandle)
-- print all values
for k, v in pairs(memoryinfo) do
    print(k.." -> "..v)
end
-- print specific value
print(memoryinfo["PagefileUsage"])
```

### Description:

This function retrieves memory usage data of the process specified by handle. Use *process.gethandle* to get handle to a process.

### Parameters:

**processHandle** - handle of process we want to get affinity for. This has to be greater than 0.

### Returns:

A hash table with memory usage information of specified process. The has table will have following keys and corresponding values stored under those keys:

- PageFaultCount
- PagefileUsage
- PeakPagefileUsage
- PeakWorkingSetSize
- PrivateUsage
- QuotaNonPagedPoolUsage
- QuotaPagedPoolUsage
- QuotaPeakNonPagedPoolUsage
- QuotaPeakPagedPoolUsage
- WorkingSetSize

I will not describe what those values mean here. You can read about those here: [http://msdn.microsoft.com/en-us/library/windows/desktop/ms684874\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684874(v=vs.85).aspx) , where **PROCESS\_MEMORY\_COUNTERS\_EX** structure is described. You will find that above keys correspond to fields in the structure.

### Throws:

Will throw exception if something goes wrong.

## process.getpriority

### Usage:

```
-- handle was previously opened by process.gethandle
priority = process.getpriority (processHandle)
```

### Description:

This function will get priority for process specified by its handle.

### Parameters:

**processHandle** - handle of process we want to get priority for. This has to be greater than 0. Use *process.gethandle* to get handle to a process.

### Returns:

Priority of the process. Either: **P\_ABOVE\_NORMAL**, **P\_BELOW\_NORMAL**, **P\_HIGH**, **P\_IDLE**, **P\_NORMAL**, or **P\_REALTIME**.

### Throws:

Will throw exception if something goes wrong.

## process.getprivatebytes

### Usage:

```
-- handle was previously opened by process.gethandle  
privatebytes = process.getprivatebytes (processHandle)
```

### Description:

This function retrieves the amount of memory private bytes the process specified by handle is using. Use *process.gethandle* to get handle to a process.

### Parameters:

**processHandle** - handle of process we want to get affinity for. This has to be greater than 0.

### Returns:

The amount of memory private bytes the process is using.

### Throws:

Will throw exception if something goes wrong.

## process.getid

### Usage:

```
-- get id for current process, ie. the one under which this script run  
pid = process.getid()  
-- get id for first process whose name matches "notepad"  
pid = process.getid("notepad")  
-- get id of process with handle 5020  
pid = process.getid(5020)
```

### Description:

This function will return process id for first process matching the given name (at least partially) or the process with given handle. If no parameters are given, the id for current process is returned. The name matching is actually “contains” behavior, so you do not have to match the name entirely to get some result.

### Parameters:

**processName** - name of the process we want to get process ID for. Can be part of name.  
**processHandle** - handle of process we want to get process ID for.

### Returns:

ID of process matching given name, given handle or ID of current process.

### Throws:

Will throw exception if something goes wrong.

Intentionally Left Blank

## process.getids

### Usage:

```
-- get ids for processes whose name matches "notepad"
pids = process.getids("notepad")
-- print all ids
for k,v in pairs(pids) do
    print(v)
end
```

### Description:

This function will return an array containing ID's of processes whose name matches (at least partially) given name. The name matching is actually “contains” behavior, so you do not have to match the name entirely to get some result.

### Parameters:

**processName** - name of the process we want to get process ID's for. Can be part of name.

### Returns:

ID's of all processes whose name matches **processName**.

### Throws:

Will throw exception if something goes wrong.

## process.setaffinity

### Usage:

```
-- set affinity for process to 1
process.setaffinity (processHandle, 0x1)
```

### Description:

This function will set affinity for process specified by its handle.

### Parameters:

**processHandle** - handle of process we want to change affinity for. This has to be greater than 0. Use *process.gethandle* to get handle to a process.

**affinity** - affinity we want to set. This must greater than 0.

### Returns:

Nothing if there were no errors. Otherwise, throws exception.

### Throws:

Will throw exception if something goes wrong.

Intentionally Left Blank

## process.setpriority

### Usage:

```
-- set priority for process to idle  
process.setpriority (processHandle, P_IDLE)
```

### Description:

This function will set priority for process specified by its handle.

### Parameters:

**processHandle** – handle of process we want to change affinity for. This has to be greater than 0. Use *process.gethandle* to get handle to a process.  
**priority** – priority the process should have. One of following constants are supported: `P_ABOVE_NORMAL`, `P_BELOW_NORMAL`, `P_HIGH`, `P_IDLE`, `P_NORMAL`, or `P_REALTIME`.

### Returns:

Nothing if there were no errors. Otherwise, throws exception.

### Throws:

Will throw exception if something goes wrong.

## 5. Functions for window manipulation

All functions for window manipulation are exported to table: **window**. Which means you have to call them like: **window.function**.

### window.find

#### Usage:

```
-- find handles for windows that has "Notepad" in title bar
whandles = window.find("Notepad")
-- find handles for windows that has "Notepad" in title bar
-- and belongs to process with id = 5020
whandles = window.find("Notepad", 5020)
-- find handles for windows that has "Notepad" in title bar
-- and are top level ones (has no parent window)
whandles = window.find("Notepad", WE_TOPLEVELONLY)
-- find handles for windows that has "Notepad" in title bar
-- and belongs to process with id = 5020 and are top level ones
-- (has no parent window)
-- find handles for windows that has "Notepad" in title bar
whandles = window.find("Notepad", 5020, WE_TOPLEVELONLY)
-- print all handles
for k,v in pairs(whandles) do
  print("Window handle: "..v)
end
```

#### Description:

This function tries to find windows specified by text in its title bar. Optionally, the matching criteria may be tightened by specifying also ID of process the window must belong to and whether the window can be any window or top level only (window that is not child of any other window – has no parent). If you will call the function with 2 arguments, the function expects that the second argument is **toplevelonly**. If you call it with three arguments, the second one is **processID** and **toplevelonly** is the third one.

#### Parameters:

**windowText** - the text of the window title bar that must be matched (partially)  
**processID** - ID of process the window must belong to.  
**toplevelonly** - Whether the window must be a top level window (not a child window).  
Supported constants are: [WE\\_ALL](#) – for all windows,  
[WE\\_TOPLEVELONLY](#) – for top-level windows only. Default is [WE\\_ALL](#).

#### Returns:

An array of window handles belonging to given process.

#### Throws:

Will throw exception if something goes wrong.

Intentionally Left Blank

## window.getclientsize

### Usage:

```
-- get client size for window with handle 145896
-- use window.find or window.gethandles to get window handle(s)
cw, ch = window.getclientsize(145896)
```

### Description:

This function returns client size of window specified by its handle.

### Parameters:

**hWnd** - handle of window we want to get client size for.

### Returns:

Width and height of the window client area.

### Throws:

Will throw exception if something goes wrong.

## window.gethandles

### Usage:

```
-- find all windows belonging to process with id = 5020
whandles = window.gethandles(pid)
-- find all top level windows belonging to process with id = 5020
whandles = window.gethandles(pid, WE_TOPLEVELONLY)
-- print what we found
for key, value in pairs(whandles) do
    print(value)
end
```

### Description:

This function returns handles for windows owned by process specified by given process id. It is possible to find all windows or only top level windows (window that is not child of any other window – has no parent).

### Parameters:

**processID** - id of process we want to find window handles for.

**toplevelonly** - Whether the window must be a top level window (not a child window).

Supported constants are: [WE\\_ALL](#) – for all windows,

[WE\\_TOPLEVELONLY](#) – for top-level windows only. Default is [WE\\_ALL](#).

### Returns:

An array of window handles belonging to given process.

### Throws:

Will throw exception if something goes wrong.

Intentionally Left Blank

## window.getsize

### Usage:

```
-- get size for window with handle 145896
-- use window.find or window.gethandles to get window handle(s)
w, h = window.getsize(145896)
```

### Description:

This function returns size of window specified by its handle.

### Parameters:

**hWnd** - handle of window we want to get size for.

### Returns:

Width and height of the window.

### Throws:

Will throw exception if something goes wrong.

## window.getstyle

### Usage:

```
-- get style for window with handle 145896
- - use window.find or window.gethandles to get window handle(s)
style = window.getstyle(145896)
```

### Description:

This function returns the style of window specified by handle.

### Parameters:

**hWnd** - handle of window we want to get style for.

### Returns:

Window style. See:

[http://msdn.microsoft.com/en-us/library/windows/desktop/ms632600\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms632600(v=vs.85).aspx) for possible values.

### Throws:

Will throw exception if something goes wrong.

## window.gettext

### Usage:

```
-- get text for window with handle 145896
-- use window.find or window.gethandles to get window handle(s)
text = window.gettext(145896)
```

### Description:

This function returns text in the title bar for window specified by handle.

### Parameters:

**hWnd** - handle of window we want to get text for.

### Returns:

A text in the title bar of specified window.

### Throws:

Will throw exception if something goes wrong.

## window.isforeground

### Usage:

```
-- check whether window with handle 145896 is foreground
-- use window.find or window.gethandles to get window handle(s)
isforeground = window.isforeground(145896)
```

### Description:

This function tests whether given window is foreground window.

### Parameters:

**hwnd** - handle of window we want to check that is foreground.

### Returns:

True if given window is at foreground, otherwise false.

### Throws:

Will throw exception if something goes wrong.

## window.isfullscreen

### Usage:

```
-- check whether window with handle 145896 is fullscreen
-- use window.find or window.gethandles to get window handle(s)
isfullscreen = window.isfullscreen(145896)
```

### Description:

This function tests whether the given window is in full screen mode – if it's size is the same or greater than the size of the monitor the window is displayed on.

### Parameters:

**hwnd** - handle of window we want to test.

### Returns:

True if window size is the same or greater than the size of the monitor it is displayed on, otherwise false.

### Throws:

Will throw exception if something goes wrong.

## window.maximize

### Usage:

```
-- maximize window with handle 145896
window.maximize(145896)
```

### Description:

This function maximizes window with given handle. The window will be maximized but not activated nor brought to foreground.

### Parameters:

**hwnd** - handle of window we want to maximize.

### Returns:

Nothing if no errors, otherwise, exception is thrown.

### Throws:

Will throw exception if something goes wrong.

## window.minimize

### Usage:

```
-- minimize window with handle 145896  
window.minimize(145896)
```

### Description:

This function minimizes the given window.

### Parameters:

**hwnd** - handle of window we want to minimize.

### Returns:

Nothing if no errors, otherwise, exception is thrown.

### Throws:

Will throw exception if something goes wrong.

## window.move

### Usage:

```
-- move window with handle 145896 to coordinates 50,50  
window.move(145896, 50, 50)  
-- move window with handle 145896 to coordinates 50,50  
-- and resize it to 400x300  
window.move(145896, 50, 50, 400, 300)
```

### Description:

This function moves and optionally re-sizes window specified by its handle.

### Parameters:

**hwnd** - handle of window we want to move.

**x** - x position of top left corner.

**y** - y position of top left corner.

**width** - width of the window

**height** - height of the window

### Returns:

Nothing if no errors, otherwise, exception is thrown.

### Throws:

Will throw exception if something goes wrong.

## window.restore

### Usage:

```
-- restore window with handle 145896  
window.restore(145896)
```

### Description:

This function restores the window specified by its handle. It activates and displays the window. If the window was maximized or minimized, it is restored to its original size and position.

### Parameters:

**hwnd** - handle of window we want to restore.

### Returns:

Nothing if no errors, otherwise, exception is thrown.

### Throws:

Will throw exception if something goes wrong.

## window.sendMessage

### Usage:

```
-- restore window with handle = 145896
local WM_SYSCOMMAND = 0x0112
local SC_RESTORE = 0xF120
window.sendMessage(145896, WM_SYSCOMMAND, SC_RESTORE, 0)
```

### Description:

This function is direct wrapper for SendMessage. See: [http://msdn.microsoft.com/en-us/library/windows/desktop/ms644950\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms644950(v=vs.85).aspx) for more details.

### Parameters:

**hWnd** - handle of window we want to send message to.  
**Msg** - the message to be sent.  
**wParam** - Additional message-specific information.  
**lParam** - Additional message-specific information.

### Returns:

The value returned by SendMessage windows function.

### Throws:

Will throw exception if something goes wrong.

## window.setforeground

### Usage:

```
-- bring window with handle 145896 to foreground.
window.setforeground(145896)
```

### Description:

This function brings selected window to foreground.

### Parameters:

**hWnd** - handle of window we want to bring foreground.

### Returns:

Nothing if no errors, otherwise, exception is thrown.

### Throws:

Will throw exception if something goes wrong.

## window.setfullscreen

### Usage:

```
-- set window with handle 145896 to full-screen mode.
window.setfullscreen(145896)
```

### Description:

This function switches given window to full-screen mode. However, this does not do any true Direct-X exclusive full-screen mode. This function will simply maximize the window and remove its borders, so it fills the whole screen.

### Parameters:

**hWnd** - handle of window we want to get focus to.

### Returns:

Nothing if no errors, otherwise, exception is thrown.

### Throws:

Will throw exception if something goes wrong.

## window.setwindowed

### Usage:

```
-- set window with handle 145896 to windowed mode.  
window.setwindowed(145896)
```

### Description:

This function switches given window to windowed mode. It changes window style so it has title bar and border. You will notice no change if the window already has a border of course. This function can be used for example to return “normal windowed” style to window maximized with *window.setfullscreen*.

### Parameters:

**hwnd** - handle of window we want to have border.

### Returns:

Nothing if no errors, otherwise, exception is thrown.

### Throws:

Will throw exception if something goes wrong.

## window.setposition

### Usage:

```
-- move window with handle=145896 to position 0,0 (left top corner of primary  
screen)  
-- and resize it to 400x200.  
window.setposition(145896, 0, 0, 0, 400, 200)  
-- move window with handle=145896 to position 0,0 (left top corner of primary  
screen)  
-- and resize it to 400x200, specifying flags  
local SWP_SHOWWINDOW = 0x0040  
window.setposition(145896, 0, 0, 0, 400, 200, SWP_SHOWWINDOW)
```

### Description:

This function is simple wrapper for windows function: SetWindowPos. See: [http://msdn.microsoft.com/en-us/library/windows/desktop/ms633545\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms633545(v=vs.85).aspx)

### Parameters:

**hwnd** - handle of window we want to get size for.

**hwndInsertAfter** - handle of window we want to move this window before in Z order.

**x** - x position of top left corner.

**y** - y position of top left corner.

**width** - width of the window

**height** - height of the window

**uFlags** - flags. See:

[http://msdn.microsoft.com/en-us/library/windows/desktop/ms633545\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms633545(v=vs.85).aspx)  
for details. If you don't specify any, **SWP\_NOACTIVATE** | **SWP\_NOZORDER** will be used.

### Returns:

Nothing if no errors, otherwise, exception is thrown.

### Throws:

Will throw exception if something goes wrong.

## window.setStyle

### Usage:

```
-- set style of window with handle=145896 to WS_POPUP
local WS_POPUP = 0x80000000
window.setStyle(145896, WS_POPUP)
-- this is the same as above
window.setStyle(145896, WS_POPUP, WSOP_REPLACE)
-- remove caption and border from window
local WS_CAPTION = 0x00C00000
window.setStyle(145896, WS_CAPTION, WSOP_REMOVE)
-- add caption and border to window
window.setStyle(145896, WS_CAPTION, WSOP_ADD)
```

### Description:

This function alters style for window specified by its handle.

### Parameters:

**hwnd** - handle of window we want to get style for.

**style** - the style we want to replace, add or remove by/to/from current style. See: [http://msdn.microsoft.com/en-us/library/windows/desktop/ms632600\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms632600(v=vs.85).aspx) for possible values.

**operation** - what to do with old style and new style. Values are:

**WSOP\_REPLACE** – replace old style with new style.

**WSOP\_ADD** – add new style to old style.

**WSOP\_REMOVE** – remove new style from old style.

### Returns:

Nothing if no errors, otherwise exception is raised.

### Throws:

Will throw exception if something goes wrong.

End Of The Document