

Forum:[Hack Forums](#)

Thread: [\[C#\]\[VB.NET\] Icon Changer - UpdateResource API \(No Reshacker or .Dll\)](#)

GoldStar - May 15, 2010, 7:24 am

i just found this class, it can change the icon of a win32 app with not 3rd party apps or dlls. it's uses the same method of iStealer for changing the icon, i though i would share. credits goes to the chinese guy who wrote it (have no idea, the source is commmented in chinese)

when i have the time i'll try to convert to vb, i hope i can do it, as i saw some pointers in the source (you know, vb.net does'nt support pointers to variables).

anyway, enjoy :

Edit : Solution Added.

btw worked good for me.

looks something like this :

[img]<http://img194.imageshack.us/img194/9619/icoo.jpg>[/img]

Download (C#) : <http://www.mediafire.com/?dzmzrz3yth4>

Edit : i was almost done, but Julien is the winner of the day xD

here is it so credits goes to him :

Code Imports System.Runtime.InteropServices

Imports System.Security

Public Class IconInjector

' Basically, you can change icons with the UpdateResource api call.

' When you make the call you say "I'm updating an icon", and you send the icon data.

' The main problem is that ICO files store the icons in one set of structures, and exe/dll files store them in

' another set of structures. So you have to translate between the two -- you can't just load the ICO file as

' bytes and send them with the UpdateResource api call.

<SuppressUnmanagedCodeSecurity(> _

Private Class NativeMethods

<DllImport("kernel32")> _

Public Shared Function BeginUpdateResource(_

ByVal fileName As String, _

<MarshalAs(UnmanagedType.Bool)> ByVal deleteExistingResources As Boolean) As IntPtr

End Function

<DllImport("kernel32")> _

Public Shared Function UpdateResource(_

ByVal hUpdate As IntPtr, _

ByVal type As IntPtr, _

ByVal name As IntPtr, _

ByVal language As Short, _

```

    <MarshalAs(UnmanagedType.LPArray, SizeParamIndex:=5)> _
    ByVal data() As Byte, _
    ByVal dataSize As Integer) As <MarshalAs(UnmanagedType.Bool)> Boolean
End Function

<DllImport("kernel32")> _
Public Shared Function EndUpdateResource( _
    ByVal hUpdate As IntPtr, _
    <MarshalAs(UnmanagedType.Bool)> ByVal discard As Boolean) As
<MarshalAs(UnmanagedType.Bool)> Boolean
End Function
End Class

' The first structure in an ICO file lets us know how many images are in the file.
<StructLayout(LayoutKind.Sequential)> _
Private Structure ICONDIR
    Public Reserved As UShort ' Reserved, must be 0
    Public Type As UShort ' Resource type, 1 for icons.
    Public Count As UShort ' How many images.
    ' The native structure has an array of ICONDIRENTRYs as a final field.
End Structure

' Each ICONDIRENTRY describes one icon stored in the ico file. The offset says where the icon
image data
' starts in the file. The other fields give the information required to turn that image data into a valid
' bitmap.
<StructLayout(LayoutKind.Sequential)> _
Private Structure ICONDIRENTRY
    Public Width As Byte ' Width, in pixels, of the image
    Public Height As Byte ' Height, in pixels, of the image
    Public ColorCount As Byte ' Number of colors in image (0 if >=8bpp)
    Public Reserved As Byte ' Reserved ( must be 0)
    Public Planes As UShort ' Color Planes
    Public BitCount As UShort ' Bits per pixel
    Public BytesInRes As Integer ' Length in bytes of the pixel data
    Public ImageOffset As Integer ' Offset in the file where the pixel data starts.
End Structure

' Each image is stored in the file as an ICONIMAGE structure:
'typedef struct
'{
' BITMAPINFOHEADER icHeader; // DIB header
' RGBQUAD icColors[1]; // Color table
' BYTE icXOR[1]; // DIB bits for XOR mask
' BYTE icAND[1]; // DIB bits for AND mask
'} ICONIMAGE, *LPICONIMAGE;

<StructLayout(LayoutKind.Sequential)> _
Private Structure BITMAPINFOHEADER
    Public Size As UInteger
    Public Width As Integer
    Public Height As Integer

```

```

Public Planes As UShort
Public BitCount As UShort
Public Compression As UInteger
Public SizeImage As UInteger
Public XPelsPerMeter As Integer
Public YPelsPerMeter As Integer
Public ClrUsed As UInteger
Public ClrImportant As UInteger
End Structure

```

' The icon in an exe/dll file is stored in a very similar structure:

```

<StructLayout(LayoutKind.Sequential, Pack:=2)> _

```

```

Private Structure GRPICONDIRENTRY

```

```

    Public Width As Byte
    Public Height As Byte
    Public ColorCount As Byte
    Public Reserved As Byte
    Public Planes As UShort
    Public BitCount As UShort
    Public BytesInRes As Integer
    Public ID As UShort

```

```

End Structure

```

```

Public Shared Sub InjectIcon(ByVal exeFileName As String, ByVal iconFileName As String)

```

```

    InjectIcon(exeFileName, iconFileName, 1, 1)

```

```

End Sub

```

```

Public Shared Sub InjectIcon(ByVal exeFileName As String, ByVal iconFileName As String,
ByVal iconGroupID As UInteger, ByVal iconBaseID As UInteger)

```

```

    Const RT_ICON = 3UI
    Const RT_GROUP_ICON = 14UI
    Dim iconFile As IconFile = iconFile.FromFile(iconFileName)
    Dim hUpdate = NativeMethods.BeginUpdateResource(exeFileName, False)
    Dim data = iconFile.CreateIconGroupData(iconBaseID)
    NativeMethods.UpdateResource(hUpdate, New IntPtr(RT_GROUP_ICON), New
IntPtr(iconGroupID), 0, data, data.Length)
    For i = 0 To iconFile.ImageCount - 1
        Dim image = iconFile.ImageData(i)
        NativeMethods.UpdateResource(hUpdate, New IntPtr(RT_ICON), New IntPtr(iconBaseID +
i), 0, image, image.Length)
    Next
    NativeMethods.EndUpdateResource(hUpdate, False)
End Sub

```

```

Private Class IconFile

```

```

    Private iconDir As New ICONDIRENTRY
    Private iconEntry() As ICONDIRENTRY
    Private iconImage()() As Byte

```

```

    Public ReadOnly Property ImageCount As Integer

```

```

        Get

```

```

            Return iconDir.Count

```

```
End Get
End Property
```

```
Public ReadOnly Property ImageData(ByVal index As Integer) As Byte()
    Get
        Return iconImage(index)
    End Get
End Property
```

```
Private Sub New()
End Sub
```

```
Public Shared Function FromFile(ByVal filename As String) As IconFile
    Dim instance As New IconFile
    ' Read all the bytes from the file.
    Dim fileBytes() As Byte = IO.File.ReadAllBytes(filename)
    ' First struct is an ICONDIR
    ' Pin the bytes from the file in memory so that we can read them.
    ' If we didn't pin them then they could move around (e.g. when the
    ' garbage collector compacts the heap)
    Dim pinnedBytes = GCHandle.Alloc(fileBytes, GCHandleType.Pinned)
    ' Read the ICONDIR
    instance.iconDir = DirectCast(Marshal.PtrToStructure(pinnedBytes.AddrOfPinnedObject,
GetType(ICONDIR)), ICONDIR)
    ' which tells us how many images are in the ico file. For each image, there's a
    ICONDIRENTRY, and associated pixel data.
    instance.iconEntry = New ICONDIRENTRY(instance.iconDir.Count - 1) {}
    instance.iconImage = New Byte(instance.iconDir.Count - 1)() {}
    ' The first ICONDIRENTRY will be immediately after the ICONDIR, so the offset to it is the
    size of ICONDIR
    Dim offset = Marshal.SizeOf(instance.iconDir)
    ' After reading an ICONDIRENTRY we step forward by the size of an
    ICONDIRENTRY
    Dim iconDirEntryType = GetType(ICONDIRENTRY)
    Dim size = Marshal.SizeOf(iconDirEntryType)
    For i = 0 To instance.iconDir.Count - 1
        ' Grab the structure.
        Dim entry = DirectCast(Marshal.PtrToStructure(New
IntPtr(pinnedBytes.AddrOfPinnedObject.ToInt64 + offset), iconDirEntryType), ICONDIRENTRY)
        instance.iconEntry(i) = entry
        ' Grab the associated pixel data.
        instance.iconImage(i) = New Byte(entry.BytesInRes - 1) {}
        Buffer.BlockCopy(fileBytes, entry.ImageOffset, instance.iconImage(i), 0,
entry.BytesInRes)
        offset += size
    Next
    pinnedBytes.Free()
    Return instance
End Function
```

```
Public Function CreateIconGroupData(ByVal iconBaseID As UInteger) As Byte()
    ' This will store the memory version of the icon.
    Dim sizeOfIconGroupData As Integer = Marshal.SizeOf(GetType(ICONDIR)) +
```

```

Marshal.SizeOf(GetType(GRPICONDIRENTRY)) * ImageCount
Dim data(sizeOfIconGroupData - 1) As Byte
Dim pinnedData = GCHandle.Alloc(data, GCHandleType.Pinned)
Marshal.StructureToPtr(iconDir, pinnedData.AddrOfPinnedObject, False)
Dim offset = Marshal.SizeOf(iconDir)
For i = 0 To ImageCount - 1
    Dim grpEntry As New GRPICONDIRENTRY
    Dim bitmapheader As New BITMAPINFOHEADER
    Dim pinnedBitmapInfoHeader = GCHandle.Alloc(bitmapheader, GCHandleType.Pinned)
    Marshal.Copy(ImageData(i), 0, pinnedBitmapInfoHeader.AddrOfPinnedObject,
Marshal.SizeOf(GetType(BITMAPINFOHEADER)))
    pinnedBitmapInfoHeader.Free()
    grpEntry.Width = iconEntry(i).Width
    grpEntry.Height = iconEntry(i).Height
    grpEntry.ColorCount = iconEntry(i).ColorCount
    grpEntry.Reserved = iconEntry(i).Reserved
    grpEntry.Planes = bitmapheader.Planes
    grpEntry.BitCount = bitmapheader.BitCount
    grpEntry.BytesInRes = iconEntry(i).BytesInRes
    grpEntry.ID = CType(iconBaseID + i, UShort)
    Marshal.StructureToPtr(grpEntry, New IntPtr(pinnedData.AddrOfPinnedObject.ToInt64 +
offset), False)
    offset += Marshal.SizeOf(GetType(GRPICONDIRENTRY))
Next
pinnedData.Free()
Return data
End Function

```

End Class

End Class

Usage :

Code IconInjector.InjectIcon(ExePath, IconPath)

DarkSeven46 - May 15, 2010, 7:26 am

Very nice share. Thanks

(2short)

08maxam - May 15, 2010, 7:27 am

Vn mate thanks for sharing with hf :) getting my post up

GoldStar - May 15, 2010, 8:05 am

bump : Solution project added.

schi - May 15, 2010, 8:14 am

Nice job man;)

too short

Odayleet - May 15, 2010, 8:34 am

Wow, great share thanks !

Sikandar - May 15, 2010, 9:14 am

coool share miharbi
thread rating added!

Cunt - May 15, 2010, 9:15 am

great share!!!! goooo chineseese!!! they pwn :P and yes thanks if u convert to vb that would be great ! :)

GoldStar - May 15, 2010, 9:17 am

(May 15, 2010 09:15 AM)SCsDecoy Wrote: [\[CITE\]](#) great share!!!! goooo chineseese!!! they pwn :P and yes thanks if u convert to vb that would be great ! :)
no problem guys

still working on converting it, but the damn vb doesn't support pointers. still trying to work around it with marshaling.

DexHacker - May 15, 2010, 9:55 am

Looks good, thanks for the share.

laptopguy - May 15, 2010, 9:58 am

Yeah this is nice thanks mate :)

Your Pal - May 15, 2010, 10:22 am

This is great.

Many thanks.

GoldStar - May 15, 2010, 2:29 pm

Bump for VB.Net Class (Reflexion)

Soul Collector - May 15, 2010, 3:16 pm

Thanks a lot mate.. 5* rated!

GoldStar - May 15, 2010, 3:20 pm

(May 15, 2010 03:15 PM)NyC.xXBRONX.NyC Wrote: [\[CITE\]](#) wait so i assume you figured it out and it works? or just unstable?

no it's stable. but i didn't literally converted it to vb, but calling the c# assembly at runtime. try it your self.

julian - May 15, 2010, 3:30 pm

If I'm understanding you right, you are just using the C# dll then, because of the pointers? I'll have a go at converting it by hand...

GoldStar - May 15, 2010, 3:32 pm

(May 15, 2010 03:30 PM)julian Wrote: [\[CITE\]](#) If I'm understanding you right, you are just using the C# dll then, because of the pointers? I'll have a go at converting it by hand...

yeah. but can't figure it out. the magic of marshal is not working this time. can't understand why. tried structuretoptr, and getpinnedobject of a gchandle... can't find the right alternative for it.

muhammad11 - May 15, 2010, 3:49 pm

thanks very much mate...

i will add credit from u...

julian - May 15, 2010, 5:15 pm

Here's a rough vb.net translation. It worked for the first test I ran, changing the icon in an old 32 bit program.

```
Code Option Strict On
Option Explicit On
Option Infer On
```

```
Imports System.Runtime.InteropServices
Imports System.Security
```

```
Public Class IconInjector
```

```
    ' Basically, you can change icons with the UpdateResource api call.
    ' When you make the call you say "I'm updating an icon", and you send the icon data.
    ' The main problem is that ICO files store the icons in one set of structures, and exe/dll files store
    them in
    ' another set of structures. So you have to translate between the two -- you can't just load the
    ICO file as
    ' bytes and send them with the UpdateResource api call.
```

```
    <SuppressUnmanagedCodeSecurity()> _
    Private Class NativeMethods
        <DllImport("kernel32")> _
        Public Shared Function BeginUpdateResource( _
            ByVal fileName As String, _
            <MarshalAs(UnmanagedType.Bool)> ByVal deleteExistingResources As Boolean) As IntPtr
        End Function

        <DllImport("kernel32")> _
        Public Shared Function UpdateResource( _
            ByVal hUpdate As IntPtr, _
            ByVal type As IntPtr, _
            ByVal name As IntPtr, _
            ByVal language As Short, _
            <MarshalAs(UnmanagedType.LPArray, SizeParamIndex:=5)> _
            ByVal data() As Byte, _
            ByVal dataSize As Integer) As <MarshalAs(UnmanagedType.Bool)> Boolean
        End Function

        <DllImport("kernel32")> _
        Public Shared Function EndUpdateResource( _
            ByVal hUpdate As IntPtr, _
            <MarshalAs(UnmanagedType.Bool)> ByVal discard As Boolean) As
        <MarshalAs(UnmanagedType.Bool)> Boolean
        End Function
    End Class
```

```
    ' The first structure in an ICO file lets us know how many images are in the file.
```

```
    <StructLayout(LayoutKind.Sequential)> _
```

```
    Private Structure ICONDIR
```

```
        Public Reserved As UShort ' Reserved, must be 0
```

```
        Public Type As UShort ' Resource type, 1 for icons.
```

```

    Public Count As UShort    ' How many images.
    ' The native structure has an array of ICONDIRENTRYs as a final field.
End Structure

```

' Each ICONDIRENTRY describes one icon stored in the ico file. The offset says where the icon image data starts in the file. The other fields give the information required to turn that image data into a valid bitmap.

```

<StructLayout(LayoutKind.Sequential)> _
Private Structure ICONDIRENTRY
    Public Width As Byte      ' Width, in pixels, of the image
    Public Height As Byte     ' Height, in pixels, of the image
    Public ColorCount As Byte  ' Number of colors in image (0 if >=8bpp)
    Public Reserved As Byte    ' Reserved ( must be 0)
    Public Planes As UShort    ' Color Planes
    Public BitCount As UShort  ' Bits per pixel
    Public BytesInRes As Integer ' Length in bytes of the pixel data
    Public ImageOffset As Integer ' Offset in the file where the pixel data starts.
End Structure

```

' Each image is stored in the file as an ICONIMAGE structure:

```

typedef struct
{
    ' BITMAPINFOHEADER icHeader;    // DIB header
    ' RGBQUAD          icColors[1]; // Color table
    ' BYTE             icXOR[1];    // DIB bits for XOR mask
    ' BYTE             icAND[1];    // DIB bits for AND mask
} ICONIMAGE, *LPICONIMAGE;

```

```

<StructLayout(LayoutKind.Sequential)> _
Private Structure BITMAPINFOHEADER
    Public Size As UInteger
    Public Width As Integer
    Public Height As Integer
    Public Planes As UShort
    Public BitCount As UShort
    Public Compression As UInteger
    Public SizeImage As UInteger
    Public XPelsPerMeter As Integer
    Public YPelsPerMeter As Integer
    Public ClrUsed As UInteger
    Public ClrImportant As UInteger
End Structure

```

' The icon in an exe/dll file is stored in a very similar structure:

```

<StructLayout(LayoutKind.Sequential, Pack:=2)> _
Private Structure GRPICONDIRENTRY
    Public Width As Byte
    Public Height As Byte
    Public ColorCount As Byte
    Public Reserved As Byte
    Public Planes As UShort

```

```
Public BitCount As UShort
Public BytesInRes As Integer
Public ID As UShort
End Structure
```

```
Public Shared Sub InjectIcon(ByVal exeFileName As String, ByVal iconFileName As String)
    InjectIcon(exeFileName, iconFileName, 1, 1)
End Sub
```

```
Public Shared Sub InjectIcon(ByVal exeFileName As String, ByVal iconFileName As String,
ByVal iconGroupID As UInteger, ByVal iconBaseID As UInteger)
    Const RT_ICON = 3UI
    Const RT_GROUP_ICON = 14UI
    Dim iconFile As IconFile = iconFile.FromFile(iconFileName)
    Dim hUpdate = NativeMethods.BeginUpdateResource(exeFileName, False)
    Dim data = iconFile.CreateIconGroupData(iconBaseID)
    NativeMethods.UpdateResource(hUpdate, New IntPtr(RT_GROUP_ICON), New
IntPtr(iconGroupID), 0, data, data.Length)
    For i = 0 To iconFile.ImageCount - 1
        Dim image = iconFile.ImageData(i)
        NativeMethods.UpdateResource(hUpdate, New IntPtr(RT_ICON), New IntPtr(iconBaseID +
i), 0, image, image.Length)
    Next
    NativeMethods.EndUpdateResource(hUpdate, False)
End Sub
```

```
Private Class IconFile
```

```
    Private iconDir As New ICONDIR
    Private iconEntry() As ICONDIRENTRY
    Private iconImage()() As Byte
```

```
    Public ReadOnly Property ImageCount As Integer
        Get
            Return iconDir.Count
        End Get
    End Property
```

```
    Public ReadOnly Property ImageData(ByVal index As Integer) As Byte()
        Get
            Return iconImage(index)
        End Get
    End Property
```

```
    Private Sub New()
    End Sub
```

```
    Public Shared Function FromFile(ByVal filename As String) As IconFile
        Dim instance As New IconFile
        ' Read all the bytes from the file.
        Dim fileBytes() As Byte = IO.File.ReadAllBytes(filename)
        ' First struct is an ICONDIR
        ' Pin the bytes from the file in memory so that we can read them.
```

```

' If we didn't pin them then they could move around (e.g. when the
' garbage collector compacts the heap)
Dim pinnedBytes = GCHandle.Alloc(fileBytes, GCHandleType.Pinned)
' Read the ICONDIR
instance.iconDir = DirectCast(Marshal.PtrToStructure(pinnedBytes.AddrOfPinnedObject,
GetType(ICONDIR)), ICONDIR)
' which tells us how many images are in the ico file. For each image, there's a
ICONDIRENTRY, and associated pixel data.
instance.iconEntry = New ICONDIRENTRY(instance.iconDir.Count - 1) {}
instance.iconImage = New Byte(instance.iconDir.Count - 1) {}
' The first ICONDIRENTRY will be immediately after the ICONDIR, so the offset to it is the
size of ICONDIR
Dim offset = Marshal.SizeOf(instance.iconDir)
' After reading an ICONDIRENTRY we step forward by the size of an
ICONDIRENTRY
Dim iconDirEntryType = GetType(ICONDIRENTRY)
Dim size = Marshal.SizeOf(iconDirEntryType)
For i = 0 To instance.iconDir.Count - 1
    ' Grab the structure.
    Dim entry = DirectCast(Marshal.PtrToStructure(New
IntPtr(pinnedBytes.AddrOfPinnedObject.ToInt64 + offset), iconDirEntryType), ICONDIRENTRY)
    instance.iconEntry(i) = entry
    ' Grab the associated pixel data.
    instance.iconImage(i) = New Byte(entry.BytesInRes - 1) {}
    Buffer.BlockCopy(fileBytes, entry.ImageOffset, instance.iconImage(i), 0,
entry.BytesInRes)
    offset += size
Next
pinnedBytes.Free()
Return instance
End Function

```

```

Public Function CreateIconGroupData(ByVal iconBaseID As UInteger) As Byte()
' This will store the memory version of the icon.
Dim sizeOfIconGroupData As Integer = Marshal.SizeOf(GetType(ICONDIR)) +
Marshal.SizeOf(GetType(GRPICONDIRENTRY)) * ImageCount
Dim data(sizeOfIconGroupData - 1) As Byte
Dim pinnedData = GCHandle.Alloc(data, GCHandleType.Pinned)
Marshal.StructureToPtr(iconDir, pinnedData.AddrOfPinnedObject, False)
Dim offset = Marshal.SizeOf(iconDir)
For i = 0 To ImageCount - 1
    Dim grpEntry As New GRPICONDIRENTRY
    Dim bitmapheader As New BITMAPINFOHEADER
    Dim pinnedBitmapInfoHeader = GCHandle.Alloc(bitmapheader, GCHandleType.Pinned)
    Marshal.Copy(ImageData(i), 0, pinnedBitmapInfoHeader.AddrOfPinnedObject,
Marshal.SizeOf(GetType(BITMAPINFOHEADER)))
    pinnedBitmapInfoHeader.Free()
    grpEntry.Width = iconEntry(i).Width
    grpEntry.Height = iconEntry(i).Height
    grpEntry.ColorCount = iconEntry(i).ColorCount
    grpEntry.Reserved = iconEntry(i).Reserved
    grpEntry.Planes = bitmapheader.Planes
    grpEntry.BitCount = bitmapheader.BitCount

```

```

        grpEntry.BytesInRes = iconEntry(i).BytesInRes
        grpEntry.ID = CType(iconBaseID + i, UShort)
        Marshal.StructureToPtr(grpEntry, New IntPtr(pinnedData.AddrOfPinnedObject.ToInt64 +
offset), False)
        offset += Marshal.SizeOf(GetType(GRPICONDIRENTRY))
    Next
    pinnedData.Free()
    Return data
End Function

```

End Class

End Class

It could be made shorter, but I guess it's not necessary for this one.

Cunt - May 15, 2010, 5:26 pm

Yeah pretty cool :) ima try to translate in bit but i guess its hard cause of pointes or w.e :D but ill try yours ^^ above :)

also any 1 else think miharbi is super awesomely sexy?

GoldStar - May 15, 2010, 5:31 pm

(May 15, 2010 05:15 PM)julian Wrote: [CITE](#) Here's a rough vb.net translation. It worked for the first test I ran, changing the icon in an old 32 bit program.

Code Option Strict On

Option Explicit On

Option Infer On

Imports System.Runtime.InteropServices

Imports System.Security

Public Class IconInjector

' Basically, you can change icons with the UpdateResource api call.

' When you make the call you say "I'm updating an icon", and you send the icon data.

' The main problem is that ICO files store the icons in one set of structures, and exe/dll files store them in

' another set of structures. So you have to translate between the two -- you can't just load the ICO file as

' bytes and send them with the UpdateResource api call.

<SuppressUnmanagedCodeSecurity()> _

Private Class NativeMethods

<DllImport("kernel32")> _

Public Shared Function BeginUpdateResource(_

ByVal fileName As String, _

<MarshalAs(UnmanagedType.Bool)> ByVal deleteExistingResources As Boolean) As IntPtr

End Function

<DllImport("kernel32")> _

Public Shared Function UpdateResource(_

ByVal hUpdate As IntPtr, _

ByVal type As IntPtr, _

```

    ByVal name As IntPtr, _
    ByVal language As Short, _
    <MarshalAs(UnmanagedType.LPArray, SizeParamIndex:=5)> _
    ByVal data() As Byte, _
    ByVal dataSize As Integer) As <MarshalAs(UnmanagedType.Bool)> Boolean
End Function

```

```

<DllImport("kernel32")> _
Public Shared Function EndUpdateResource( _
    ByVal hUpdate As IntPtr, _
    <MarshalAs(UnmanagedType.Bool)> ByVal discard As Boolean) As
<MarshalAs(UnmanagedType.Bool)> Boolean
End Function
End Class

```

' The first structure in an ICO file lets us know how many images are in the file.

```

<StructLayout(LayoutKind.Sequential)> _
Private Structure ICONDIR

```

```

    Public Reserved As UShort ' Reserved, must be 0
    Public Type As UShort ' Resource type, 1 for icons.
    Public Count As UShort ' How many images.

```

' The native structure has an array of ICONDIRENTRYs as a final field.

```

End Structure

```

' Each ICONDIRENTRY describes one icon stored in the ico file. The offset says where the icon image data

' starts in the file. The other fields give the information required to turn that image data into a valid ' bitmap.

```

<StructLayout(LayoutKind.Sequential)> _
Private Structure ICONDIRENTRY

```

```

    Public Width As Byte ' Width, in pixels, of the image
    Public Height As Byte ' Height, in pixels, of the image
    Public ColorCount As Byte ' Number of colors in image (0 if >=8bpp)
    Public Reserved As Byte ' Reserved ( must be 0)
    Public Planes As UShort ' Color Planes
    Public BitCount As UShort ' Bits per pixel
    Public BytesInRes As Integer ' Length in bytes of the pixel data
    Public ImageOffset As Integer ' Offset in the file where the pixel data starts.

```

```

End Structure

```

' Each image is stored in the file as an ICONIMAGE structure:

```

'typdef struct
'{
' BITMAPINFOHEADER icHeader; // DIB header
' RGBQUAD icColors[1]; // Color table
' BYTE icXOR[1]; // DIB bits for XOR mask
' BYTE icAND[1]; // DIB bits for AND mask
'} ICONIMAGE, *LPICONIMAGE;

```

```

<StructLayout(LayoutKind.Sequential)> _
Private Structure BITMAPINFOHEADER
    Public Size As UInteger

```

```

Public Width As Integer
Public Height As Integer
Public Planes As UShort
Public BitCount As UShort
Public Compression As UInteger
Public SizeImage As UInteger
Public XPelsPerMeter As Integer
Public YPelsPerMeter As Integer
Public ClrUsed As UInteger
Public ClrImportant As UInteger
End Structure

```

' The icon in an exe/dll file is stored in a very similar structure:

```

<StructLayout(LayoutKind.Sequential, Pack:=2)> _

```

```

Private Structure GRPICONDIRENTRY

```

```

    Public Width As Byte
    Public Height As Byte
    Public ColorCount As Byte
    Public Reserved As Byte
    Public Planes As UShort
    Public BitCount As UShort
    Public BytesInRes As Integer
    Public ID As UShort

```

```

End Structure

```

```

Public Shared Sub InjectIcon(ByVal exeFileName As String, ByVal iconFileName As String)
    InjectIcon(exeFileName, iconFileName, 1, 1)
End Sub

```

```

Public Shared Sub InjectIcon(ByVal exeFileName As String, ByVal iconFileName As String,
ByVal iconGroupID As UInteger, ByVal iconBaseID As UInteger)
    Const RT_ICON = 3UI
    Const RT_GROUP_ICON = 14UI
    Dim iconFile As IconFile = iconFile.FromFile(iconFileName)
    Dim hUpdate = NativeMethods.BeginUpdateResource(exeFileName, False)
    Dim data = iconFile.CreateIconGroupData(iconBaseID)
    NativeMethods.UpdateResource(hUpdate, New IntPtr(RT_GROUP_ICON), New
IntPtr(iconGroupID), 0, data, data.Length)
    For i = 0 To iconFile.ImageCount - 1
        Dim image = iconFile.ImageData(i)
        NativeMethods.UpdateResource(hUpdate, New IntPtr(RT_ICON), New IntPtr(iconBaseID +
i), 0, image, image.Length)
    Next
    NativeMethods.EndUpdateResource(hUpdate, False)
End Sub

```

```

Private Class IconFile

```

```

    Private iconDir As New ICONDIRENTRY
    Private iconEntry() As ICONDIRENTRY
    Private iconImage()() As Byte

```

```

    Public ReadOnly Property ImageCount As Integer

```

```

    Get
        Return iconDir.Count
    End Get
End Property

Public ReadOnly Property ImageData(ByVal index As Integer) As Byte()
    Get
        Return iconImage(index)
    End Get
End Property

Private Sub New()
End Sub

Public Shared Function FromFile(ByVal filename As String) As IconFile
    Dim instance As New IconFile
    ' Read all the bytes from the file.
    Dim fileBytes() As Byte = IO.File.ReadAllBytes(filename)
    ' First struct is an ICONDIR
    ' Pin the bytes from the file in memory so that we can read them.
    ' If we didn't pin them then they could move around (e.g. when the
    ' garbage collector compacts the heap)
    Dim pinnedBytes = GCHandle.Alloc(fileBytes, GCHandleType.Pinned)
    ' Read the ICONDIR
    instance.iconDir = DirectCast(Marshal.PtrToStructure(pinnedBytes.AddrOfPinnedObject,
GetType(ICONDIR)), ICONDIR)
    ' which tells us how many images are in the ico file. For each image, there's a
    ICONDIRENTRY, and associated pixel data.
    instance.iconEntry = New ICONDIRENTRY(instance.iconDir.Count - 1) {}
    instance.iconImage = New Byte(instance.iconDir.Count - 1) {}
    ' The first ICONDIRENTRY will be immediately after the ICONDIR, so the offset to it is the
    size of ICONDIR
    Dim offset = Marshal.SizeOf(instance.iconDir)
    ' After reading an ICONDIRENTRY we step forward by the size of an
    ICONDIRENTRY
    Dim iconDirEntryType = GetType(ICONDIRENTRY)
    Dim size = Marshal.SizeOf(iconDirEntryType)
    For i = 0 To instance.iconDir.Count - 1
        ' Grab the structure.
        Dim entry = DirectCast(Marshal.PtrToStructure(New
IntPtr(pinnedBytes.AddrOfPinnedObject.ToInt64 + offset), iconDirEntryType), ICONDIRENTRY)
        instance.iconEntry(i) = entry
        ' Grab the associated pixel data.
        instance.iconImage(i) = New Byte(entry.BytesInRes - 1) {}
        Buffer.BlockCopy(fileBytes, entry.ImageOffset, instance.iconImage(i), 0,
entry.BytesInRes)
        offset += size
    Next
    pinnedBytes.Free()
    Return instance
End Function

Public Function CreateIconGroupData(ByVal iconBaseID As UInteger) As Byte()

```

```

' This will store the memory version of the icon.
Dim sizeOfIconGroupData As Integer = Marshal.SizeOf(GetType(ICONDIR)) +
Marshal.SizeOf(GetType(GRPICONDIRENTRY)) * ImageCount
Dim data(sizeOfIconGroupData - 1) As Byte
Dim pinnedData = GCHandle.Alloc(data, GCHandleType.Pinned)
Marshal.StructureToPtr(iconDir, pinnedData.AddrOfPinnedObject, False)
Dim offset = Marshal.SizeOf(iconDir)
For i = 0 To ImageCount - 1
    Dim grpEntry As New GRPICONDIRENTRY
    Dim bitmapheader As New BITMAPINFOHEADER
    Dim pinnedBitmapInfoHeader = GCHandle.Alloc(bitmapheader, GCHandleType.Pinned)
    Marshal.Copy(ImageData(i), 0, pinnedBitmapInfoHeader.AddrOfPinnedObject,
Marshal.SizeOf(GetType(BITMAPINFOHEADER)))
    pinnedBitmapInfoHeader.Free()
    grpEntry.Width = iconEntry(i).Width
    grpEntry.Height = iconEntry(i).Height
    grpEntry.ColorCount = iconEntry(i).ColorCount
    grpEntry.Reserved = iconEntry(i).Reserved
    grpEntry.Planes = bitmapheader.Planes
    grpEntry.BitCount = bitmapheader.BitCount
    grpEntry.BytesInRes = iconEntry(i).BytesInRes
    grpEntry.ID = CType(iconBaseID + i, UShort)
    Marshal.StructureToPtr(grpEntry, New IntPtr(pinnedData.AddrOfPinnedObject.ToInt64 +
offset), False)
    offset += Marshal.SizeOf(GetType(GRPICONDIRENTRY))
Next
pinnedData.Free()
Return data
End Function

End Class

```

End Class

It could be made shorter, but I guess it's not necessary for this one.

ahhh i was so close xD i missed casting and was getting the ICONDIR wrong. but i wasn't that far :p
good job btw. you're goood.

well everyone can use this instead of that shitty dll caller ^^

Edit : One More Mistake xD i declare a new object for a structure to do the sizeOf, forgot getting the type O.o

you're fast mate ^^ that would take me probably the hole night to figure it out lol

SkeetMads - May 15, 2010, 5:39 pm

Wow, really great! Thanks for sharing :)

CometJack - May 15, 2010, 5:45 pm

Where's the C# code?

I know it can be converted, but I was expecting it to be converted already via the thread title.

GoldStar - May 15, 2010, 5:47 pm

(May 15, 2010 05:45 PM)CometJack Wrote: [\[CITE\]](#) Where's the C# code?

I know it can be converted, but I was expecting it to be converted already via the thread title.
i removed it since it's in teh download. yeah it can be converted, it's already done.

jaikrishna - May 15, 2010, 8:06 pm

Very Nice Share Thanks.

But I think the same api can be used to change the assembly info. Please correct me if I'm wrong

julian - May 16, 2010, 1:07 am

It can change any resources in a PE file. I think that the xml that records assembly information does appear as a PE file resource. There are probably easier ways to do it. You can use the api to delete all the resources, which will include assembly info. There are free programs that let you browse a PE file to see what resources there are inside (e.g. [PEBrowse](#)).

?BlueAce™ - May 16, 2010, 2:03 am

gr8 share.. thnx ?

Bitwin - May 16, 2010, 2:05 am

Nice share, and thanks Julian for converting it to VB

GoldStar - May 16, 2010, 5:17 am

(May 15, 2010 08:06 PM)jaikrishna Wrote: [\[CITE\]](#) Very Nice Share Thanks.

But I think the same api can be used to change the assembly info. Please correct me if I'm wrong
you can change alot of things with this api.

[http://msdn.microsoft.com/en-us/library/ms648009\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms648009(VS.85).aspx)

P4d1n0r - May 16, 2010, 6:46 am

Very great tool mate , thanks for share

KingCobraXD - May 24, 2010, 3:52 am

Does it corrupt servers????

50GTB - May 24, 2010, 4:37 am

their ent no c# source...

Numair - May 24, 2010, 5:01 am

Thanks i will use in my new version of keylogger ofcourse i will give credits too.

GoldStar - May 28, 2010, 12:41 pm

bump

this is what can do that amazing api also.

xBroak - May 28, 2010, 1:56 pm

Anyone know how this can be adapted to show the original files icon in a picture box before selecting the new icon?

SIM-478284 - October 2, 2010, 6:19 am

(May 24, 2010 03:52 AM)KingCobraXD Wrote: [\[CITE\]](#) Does it corrupt servers????

It corrupted my server. :/

U²D - October 2, 2010, 8:09 am

This great share, you are great bro .. thx .

PlatinumHex - November 19, 2010, 3:46 pm

(Oct 2, 2010 06:19 AM)Scripter4ever Wrote: [\[CITE\]](#) It corrupted my server. :/
Same. Help would be appreciated. :)

SevX - November 19, 2010, 4:07 pm

Very nice,simple GUI...

Thanks !!!

PlatinumHex - November 19, 2010, 5:26 pm

It's not Chinese! This source is commented in Japanese. ^^

.bLaZeD - November 19, 2010, 10:32 pm

(May 15, 2010 08:05 AM)Miharbi-DoNo Wrote: [\[CITE\]](#) : Solution project added.Don't say 'b.ump.'
Make it like I did 'b.ump' Or else mods will give you 10 points of warning like me. :P

Coachip2.0 - November 25, 2010, 8:20 am

(May 15, 2010 07:24 AM)Miharbi-DoNo Wrote: [\[CITE\]](#) i just found this class, it can change the icon of a win32 app with not 3rd party apps or dlls. it's uses the same method of iStealer for changing the icon, i though i would share. credits goes to the chinese guy who wrote it (have no idea, the source is commmented in chinese)

when i have the time i'll try to convert to vb, i hope i can do it, as i saw some pointers in the source (you know, vb.net does'nt support pointers to variables).

anyway, enjoy :

Edit : Solution Added.

btw worked good for me.

looks something like this :

[img]<http://img194.imageshack.us/img194/9619/icoo.jpg>[/img]

Download (C#) : <http://www.mediafire.com/?dzmzrz3yth4>

Edit : i was almost done, but Julien is the winner of the day xD

here is it so credits goes to him :

Code Imports System.Runtime.InteropServices

Imports System.Security

Public Class IconInjector

' Basically, you can change icons with the UpdateResource api call.

' When you make the call you say "I'm updating an icon", and you send the icon data.

' The main problem is that ICO files store the icons in one set of structures, and exe/dll files store them in

' another set of structures. So you have to translate between the two -- you can't just load the ICO file as

' bytes and send them with the UpdateResource api call.

<SuppressUnmanagedCodeSecurity()> _

```

Private Class NativeMethods
<DllImport("kernel32")> _
Public Shared Function BeginUpdateResource( _
    ByVal fileName As String, _
    <MarshalAs(UnmanagedType.Bool)> ByVal deleteExistingResources As Boolean) As IntPtr
End Function

<DllImport("kernel32")> _
Public Shared Function UpdateResource( _
    ByVal hUpdate As IntPtr, _
    ByVal type As IntPtr, _
    ByVal name As IntPtr, _
    ByVal language As Short, _
    <MarshalAs(UnmanagedType.LPArray, SizeParamIndex:=5)> _
    ByVal data() As Byte, _
    ByVal dataSize As Integer) As <MarshalAs(UnmanagedType.Bool)> Boolean
End Function

<DllImport("kernel32")> _
Public Shared Function EndUpdateResource( _
    ByVal hUpdate As IntPtr, _
    <MarshalAs(UnmanagedType.Bool)> ByVal discard As Boolean) As
<MarshalAs(UnmanagedType.Bool)> Boolean
End Function
End Class

' The first structure in an ICO file lets us know how many images are in the file.
<StructLayout(LayoutKind.Sequential)> _
Private Structure ICONDIR
Public Reserved As UShort ' Reserved, must be 0
Public Type As UShort ' Resource type, 1 for icons.
Public Count As UShort ' How many images.
' The native structure has an array of ICONDIRENTRYs as a final field.
End Structure

' Each ICONDIRENTRY describes one icon stored in the ico file. The offset says where the icon
image data
' starts in the file. The other fields give the information required to turn that image data into a valid
' bitmap.
<StructLayout(LayoutKind.Sequential)> _
Private Structure ICONDIRENTRY
Public Width As Byte ' Width, in pixels, of the image
Public Height As Byte ' Height, in pixels, of the image
Public ColorCount As Byte ' Number of colors in image (0 if >=8bpp)
Public Reserved As Byte ' Reserved ( must be 0)
Public Planes As UShort ' Color Planes
Public BitCount As UShort ' Bits per pixel
Public BytesInRes As Integer ' Length in bytes of the pixel data
Public ImageOffset As Integer ' Offset in the file where the pixel data starts.
End Structure

' Each image is stored in the file as an ICONIMAGE structure:
'typdef struct

```

```
{
' BITMAPINFOHEADER icHeader; // DIB header
' RGBQUAD icColors[1]; // Color table
' BYTE icXOR[1]; // DIB bits for XOR mask
' BYTE icAND[1]; // DIB bits for AND mask
'} ICONIMAGE, *LPICONIMAGE;
```

```
<StructLayout(LayoutKind.Sequential)> _
Private Structure BITMAPINFOHEADER
Public Size As UInteger
Public Width As Integer
Public Height As Integer
Public Planes As UShort
Public BitCount As UShort
Public Compression As UInteger
Public SizeImage As UInteger
Public XPelsPerMeter As Integer
Public YPelsPerMeter As Integer
PublicClrUsed As UInteger
Public ClrImportant As UInteger
End Structure
```

' The icon in an exe/dll file is stored in a very similar structure:

```
<StructLayout(LayoutKind.Sequential, Pack:=2)> _
Private Structure GRPICONDIRENTRY
Public Width As Byte
Public Height As Byte
Public ColorCount As Byte
Public Reserved As Byte
Public Planes As UShort
Public BitCount As UShort
Public BytesInRes As Integer
Public ID As UShort
End Structure
```

```
Public Shared Sub InjectIcon(ByVal exeFileName As String, ByVal iconFileName As String)
InjectIcon(exeFileName, iconFileName, 1, 1)
End Sub
```

```
Public Shared Sub InjectIcon(ByVal exeFileName As String, ByVal iconFileName As String,
ByVal iconGroupID As UInteger, ByVal iconBaseID As UInteger)
Const RT_ICON = 3UI
Const RT_GROUP_ICON = 14UI
Dim iconFile As IconFile = iconFile.FromFile(iconFileName)
Dim hUpdate = NativeMethods.BeginUpdateResource(exeFileName, False)
Dim data = iconFile.CreateIconGroupData(iconBaseID)
NativeMethods.UpdateResource(hUpdate, New IntPtr(RT_GROUP_ICON), New
IntPtr(iconGroupID), 0, data, data.Length)
For i = 0 To iconFile.ImageCount - 1
Dim image = iconFile.ImageData(i)
NativeMethods.UpdateResource(hUpdate, New IntPtr(RT_ICON), New IntPtr(iconBaseID + i), 0,
image, image.Length)
Next
```

```
NativeMethods.EndUpdateResource(hUpdate, False)
End Sub
```

```
Private Class IconFile
```

```
Private iconDir As New ICONDIR
Private iconEntry() As ICONDIRENTRY
Private iconImage() As Byte
```

```
Public ReadOnly Property ImageCount As Integer
Get
Return iconDir.Count
End Get
End Property
```

```
Public ReadOnly Property ImageData(ByVal index As Integer) As Byte()
Get
Return iconImage(index)
End Get
End Property
```

```
Private Sub New()
End Sub
```

```
Public Shared Function FromFile(ByVal filename As String) As IconFile
Dim instance As New IconFile
' Read all the bytes from the file.
Dim fileBytes() As Byte = IO.File.ReadAllBytes(filename)
' First struct is an ICONDIR
' Pin the bytes from the file in memory so that we can read them.
' If we didn't pin them then they could move around (e.g. when the
' garbage collector compacts the heap)
Dim pinnedBytes = GCHandle.Alloc(fileBytes, GCHandleType.Pinned)
' Read the ICONDIR
instance.iconDir = DirectCast(Marshal.PtrToStructure(pinnedBytes.AddrOfPinnedObject,
GetType(ICONDIR)), ICONDIR)
' which tells us how many images are in the ico file. For each image, there's a ICONDIRENTRY,
and associated pixel data.
instance.iconEntry = New ICONDIRENTRY(instance.iconDir.Count - 1) {}
instance.iconImage = New Byte(instance.iconDir.Count - 1)() {}
' The first ICONDIRENTRY will be immediately after the ICONDIR, so the offset to it is the size of
ICONDIR
Dim offset = Marshal.SizeOf(instance.iconDir)
' After reading an ICONDIRENTRY we step forward by the size of an ICONDIRENTRY
Dim iconDirEntryType = GetType(ICONDIRENTRY)
Dim size = Marshal.SizeOf(iconDirEntryType)
For i = 0 To instance.iconDir.Count - 1
' Grab the structure.
Dim entry = DirectCast(Marshal.PtrToStructure(New
IntPtr(pinnedBytes.AddrOfPinnedObject.ToInt64 + offset), iconDirEntryType), ICONDIRENTRY)
instance.iconEntry(i) = entry
' Grab the associated pixel data.
instance.iconImage(i) = New Byte(entry.BytesInRes - 1) {}
```

```

Buffer.BlockCopy(fileBytes, entry.ImageOffset, instance.iconImage(i), 0, entry.BytesInRes)
offset += size
Next
pinnedBytes.Free()
Return instance
End Function

```

```

Public Function CreateIconGroupData(ByVal iconBaseID As UInteger) As Byte()
' This will store the memory version of the icon.
Dim sizeOfIconGroupData As Integer = Marshal.SizeOf(GetType(ICONDIR)) +
Marshal.SizeOf(GetType(GRPICONDIRENTRY)) * ImageCount
Dim data(sizeOfIconGroupData - 1) As Byte
Dim pinnedData = GCHandle.Alloc(data, GCHandleType.Pinned)
Marshal.StructureToPtr(iconDir, pinnedData.AddrOfPinnedObject, False)
Dim offset = Marshal.SizeOf(iconDir)
For i = 0 To ImageCount - 1
Dim grpEntry As New GRPICONDIRENTRY
Dim bitmapheader As New BITMAPINFOHEADER
Dim pinnedBitmapInfoHeader = GCHandle.Alloc(bitmapheader, GCHandleType.Pinned)
Marshal.Copy(ImageData(i), 0, pinnedBitmapInfoHeader.AddrOfPinnedObject,
Marshal.SizeOf(GetType(BITMAPINFOHEADER)))
pinnedBitmapInfoHeader.Free()
grpEntry.Width = iconEntry(i).Width
grpEntry.Height = iconEntry(i).Height
grpEntry.ColorCount = iconEntry(i).ColorCount
grpEntry.Reserved = iconEntry(i).Reserved
grpEntry.Planes = bitmapheader.Planes
grpEntry.BitCount = bitmapheader.BitCount
grpEntry.BytesInRes = iconEntry(i).BytesInRes
grpEntry.ID = CType(iconBaseID + i, UShort)
Marshal.StructureToPtr(grpEntry, New IntPtr(pinnedData.AddrOfPinnedObject.ToInt64 + offset),
False)
offset += Marshal.SizeOf(GetType(GRPICONDIRENTRY))
Next
pinnedData.Free()
Return data
End Function

```

End Class

End Class

Usage :

Code IconInjector.InjectIcon(ExePath, IconPath)

I added 2 function to this source:

Code try

```

{

DSStdInstallerCompiler.IconInjector.InjectIcon(textBox1.Text, textBox2.Text);

}
catch
{
}
}

```

```
finally
{
    MessageBox.Show("Icon Changed");
    Application.Exit();
}

}
```

So when the program change the icon,first show a confirm message and than close the apps.
Anyway very very good share ?

SIM-478284 - November 25, 2010, 9:40 am

(Nov 19, 2010 04:46 PM)PlatinumHex Wrote: [\[CITE\]](#) Same. Help would be appreciated. :)

I found out the problem. The icon changer does not support EOF. That is why. I made my own so I can use it to my files with EOF. :)

GoldStar - November 25, 2010, 10:16 am

(Nov 25, 2010 10:40 AM)Scripter4ever Wrote: [\[CITE\]](#) I found out the problem. The icon changer does not support EOF. That is why. I made my own so I can use it to my files with EOF. :)
you just change icon first thing. and add eof/resources later

Lambasoft - February 27, 2012, 9:34 am

Thanks a lot ! it have been hours working on !

Bay_Easy - May 8, 2012, 9:31 am

Looks great, im going to give it a try.

x64 - May 8, 2012, 12:16 pm

Maybe i'm a noob @ this , but what is the difference if I add a option too the codedom compiler our use this code instead?

Vuln - May 8, 2012, 12:33 pm

(May 8, 2012 12:16 PM)Just1c3 Wrote: [\[CITE\]](#) Maybe i'm a noob @ this , but what is the difference if I add a option too the codedom compiler our use this code instead?

This could change the icon of any -win32- file, not just the output file. If you're using CodeDOM just stick to /win32icon

x64 - May 8, 2012, 3:41 pm

(May 8, 2012 12:33 PM)[He]isenberg Wrote: [\[CITE\]](#) This could change the icon of any -win32- file, not just the output file. If you're using CodeDOM just stick to /win32icon

Ah thanks for explaining me !

I hold it by codedom ?

iownmyrat - December 3, 2012, 1:33 pm

Could you guys please update the download link?

schaai - December 16, 2012, 8:56 am

thank you for the post! this one which i was waiting for! thanks again!