

Forum:[Hack Forums](#)

Thread: [Easy Icons extractor.](#)

0xBB - January 28, 2015, 8:12 pm

Hi,

Recently someone asked me for this so I've dugged up the icon struct and found this very interesting paper :

<https://msdn.microsoft.com/en-us/library/ms997538.aspx>

So at first I declared the struct as defined here. Tho marshalling the data didn't go as expected. So I decided to proceed differently and fill em one by one using the size of the previous to write all to a stream in order to slowly build toward my icon. As you might know an icon is a "directory" of images. So it's made of few layers and it's why when it come to save you need to preserve all the struct data. What VS seems to do is convert the ICON to a BMP (notice how similar the struct are aside BMP have one layer) so the icon quality decrease. Well that said, ive spent few hours coding this lib here to ease icon extration/save(it extract all icons into a list of icon) :

If you use , please give credits.

Code Imports System.Runtime.InteropServices

Public Class Natives

'Credits: ibennz, msdn

'Version: 1.0.0.0

<DllImport("kernel32.dll", CharSet:=CharSet.Unicode)>

Private Shared Function FindResource(hModule As IntPtr, lpName As IntPtr, lpType As IntPtr)

As IntPtr

End Function

<DllImport("kernel32.dll", CharSet:=CharSet.Unicode)>

Private Shared Function FindResource(hModule As IntPtr, lpName As String, lpType As IntPtr)

As IntPtr

End Function

<DllImport("kernel32.dll", CharSet:=CharSet.Unicode)>

Private Shared Function LockResource(hResData As IntPtr) As IntPtr

End Function

<DllImport("kernel32.dll")>

Private Shared Function LoadLibraryEx(lpFileName As String, hReservedNull As IntPtr, dwFlags

As LoadLibraryFlags) As IntPtr

End Function

<System.Flags(>

Enum LoadLibraryFlags As UInteger

DONT_RESOLVE_DLL_REFERENCES = &H1

LOAD_IGNORE_CODE_AUTHZ_LEVEL = &H10

LOAD_LIBRARY_AS_DATAFILE = &H2

LOAD_LIBRARY_AS_DATAFILE_EXCLUSIVE = &H40

LOAD_LIBRARY_AS_IMAGE_RESOURCE = &H20

LOAD_WITH_ALTERED_SEARCH_PATH = &H8

End Enum

<DllImport("kernel32.dll", CharSet:=CharSet.Unicode)>

Private Shared Function SizeofResource(ByVal hModule As IntPtr, ByVal hResInfo As IntPtr) As Integer

End Function

<DllImport("kernel32", CharSet:=CharSet.Unicode)>

Private Shared Function LoadResource(ByVal hModule As IntPtr, ByVal hResInfo As IntPtr) As IntPtr

End Function

<DllImport("kernel32.dll", CharSet:=CharSet.Unicode)> _

Private Shared Function EnumResourceNames(ByVal hModule As IntPtr, ByVal dwID As Integer, ByVal lpEnumFunc As EnumResNameProcDelegate, ByVal lParam As IntPtr) As Boolean

End Function

Private Delegate Function EnumResNameProcDelegate(ByVal hModule As IntPtr, ByVal lpzType As IntPtr, ByVal lpzName As IntPtr, ByVal lParam As IntPtr) As Boolean

Public Enum ResourceType

CURSOR = 1

BITMAP = 2

ICON = 3

MENU = 4

DIALOG = 5

[STRING] = 6

FONTDIR = 7

FONT = 8

ACCELERATOR = 9

RCDATA = 10

MESSAGETABLE = 11

GROUP_CURSOR = 12

GROUP_ICON = 14

VERSION = 16

DLGINCLUDE = 17

PLUGPLAY = 19

VXD = 20

ANICURSOR = 21

ANIICON = 22

HTML = 23

MANIFEST = 24

End Enum

<StructLayout(LayoutKind.Sequential, size:=16)>

Structure IconDirEntry

Public bWidth As Byte

Public bHeight As Byte

Public bColorCount As Byte

Public bReserved As Byte

Public wPlanes As UShort

Public wBitCount As UShort

Public dwBytesInRes As UInteger

Public dwImageOffset As UInteger

End Structure

<StructLayout(LayoutKind.Sequential, size:=14)>

```

Structure GroupIconDirEntry
    Public Width As Byte
    Public Height As Byte
    Public ColorCount As Byte
    Public Reserved As Byte
    Public Planes As Int16
    Public BitCount As Int16
    Public ByteInRes As Int32
    Public nID As Int16
End Structure

```

```

<StructLayout(LayoutKind.Sequential, size:=6)>

```

```

Structure GRPICONDIR
    Public Reserved As Short
    Public Type As Short
    Public Count As Short
End Structure

```

```

<StructLayout(LayoutKind.Sequential, size:=6)>

```

```

Structure ICONDIR
    Public Reserved As UShort
    Public Type As UShort
    Public Count As UShort
End Structure

```

```

Structure TypedObject
    Public MainType As IntPtr
    Public OpType As String
    Sub New(mt As IntPtr, Optional opt As String = Nothing)
        MainType = mt
        opt = opt
    End Sub
End Structure

```

```

Private Shared PtrToIconList As New List(Of TypedObject)
Private Shared IconList As New List(Of Icon)

```

```

Public Shared Function GetIcons(ByVal FilePath As String) As List(Of Icon)

```

```

    Dim hModule As IntPtr = LoadLibraryEx(FilePath, IntPtr.Zero,
LoadLibraryFlags.LOAD_LIBRARY_AS_DATAFILE)
    If EnumResourceNames(hModule, 14, AddressOf EnumResNameProc, IntPtr.Zero) = False
Then
        Throw New Exception("No icon Found")
    End If
    For Each IconGroup As TypedObject In PtrToIconList

        Dim loc As IntPtr = FindResource(hModule, IconGroup.MainType, New IntPtr(14))
        If IconGroup.MainType = IntPtr.Zero Then
            loc = FindResource(hModule, IconGroup.OpType, New IntPtr(14))
        End If
        Dim hRes As IntPtr = LoadResource(hModule, loc)
        Dim GrpPtrlco As IntPtr = LockResource(hRes)
        Dim ResSize As Integer = SizeofResource(hModule, loc)

        Dim resBytes As Byte() = New Byte(ResSize - 1) {}
    End For
End Function

```

```

Marshal.Copy(GrpPtrlco, resBytes, 0, resBytes.Length)

Dim mst As IO.MemoryStream = New IO.MemoryStream(resBytes)

Using dst As New IO.MemoryStream()

    Dim IconHeadSize As Integer = Marshal.SizeOf(GetType(GRPICONDIR))
    Dim IconHeadBuffer As Byte() = New Byte(IconHeadSize) {}
    mst.Read(IconHeadBuffer, 0, IconHeadSize)
    Dim PtrIconhead As IntPtr = Marshal.AllocHGlobal(IconHeadSize)
    Marshal.Copy(IconHeadBuffer, 0, PtrIconhead, IconHeadSize)

    Dim IconHead As GRPICONDIR = Marshal.PtrToStructure(PtrIconhead,
GetType(GRPICONDIR))
    Marshal.FreeHGlobal(PtrIconhead)

    Dim Entries As Integer = IconHead.Count
    Dim iconImageOffset As Integer = Marshal.SizeOf(GetType(ICONDIR)) + Entries *
Marshal.SizeOf(GetType(IconDirEntry))

    Dim IconDir As New ICONDIR
    IconDir.Reserved = IconHead.Reserved
    IconDir.Type = IconHead.Type
    IconDir.Count = IconHead.Count
    Dim IconDirSize As Integer = Marshal.SizeOf(GetType(ICONDIR))
    Dim IconDirBuffer As Byte() = New Byte(IconDirSize) {}
    Dim PtrIconDir As IntPtr = Marshal.AllocHGlobal(IconDirSize)
    Marshal.StructureToPtr(IconDir, PtrIconDir, True)
    Marshal.Copy(PtrIconDir, IconDirBuffer, 0, IconDirSize)

    dst.Write(IconDirBuffer, 0, IconDirSize)

    For I As Integer = 0 To Entries - 1
        Dim GrpEntriesSize As Integer = Marshal.SizeOf(GetType(GroupIconDirEntry))
        Dim GrpEntriesBuff As Byte() = New Byte(GrpEntriesSize) {}
        mst.Read(GrpEntriesBuff, 0, GrpEntriesSize)
        Dim GrpEntriesPtr As IntPtr = Marshal.AllocHGlobal(GrpEntriesSize)
        Marshal.Copy(GrpEntriesBuff, 0, GrpEntriesPtr, GrpEntriesSize)
        Dim GrpEntires As GroupIconDirEntry = Marshal.PtrToStructure(GrpEntriesPtr,
GetType(GroupIconDirEntry))
        Marshal.FreeHGlobal(GrpEntriesPtr)

        dst.Seek(IconDirSize + I * Marshal.SizeOf(GetType(IconDirEntry)), IO.SeekOrigin.Begin)

        Dim IconDirEntry As New IconDirEntry
        IconDirEntry.bWidth = GrpEntires.Width
        IconDirEntry.bHeight = GrpEntires.Height
        IconDirEntry.bColorCount = GrpEntires.ColorCount
        IconDirEntry.bReserved = GrpEntires.Reserved
        IconDirEntry.wPlanes = GrpEntires.Planes
        IconDirEntry.wBitCount = GrpEntires.BitCount
        IconDirEntry.dwBytesInRes = GrpEntires.BytesInRes
        IconDirEntry.dwImageOffset = iconImageOffset
    
```

```

Dim DirEntrySize As Integer = Marshal.SizeOf(GetType(IconDirEntry))
Dim DirEntryBuffer As Byte() = New Byte(DirEntrySize) {}
Dim DirEntryPtr As IntPtr = Marshal.AllocHGlobal(DirEntrySize)
Marshal.StructureToPtr(IconDirEntry, DirEntryPtr, True)
Marshal.Copy(DirEntryPtr, DirEntryBuffer, 0, DirEntrySize)

```

```

dst.Write(DirEntryBuffer, 0, DirEntrySize)

```

```

'get the icon data

```

```

Dim IcoLoc As IntPtr = FindResource(hModule, New IntPtr(GrpEntires.nID), New IntPtr(3))
Dim IcoRes As IntPtr = LoadResource(hModule, IcoLoc)
Dim IcoPtr As IntPtr = LockResource(IcoRes)
Dim IcoSize As Integer = SizeofResource(hModule, IcoLoc)
Dim ImgBuff As Byte() = New Byte(IcoSize) {}
Marshal.Copy(IcoPtr, ImgBuff, 0, IcoSize)

```

```

dst.Seek(iconImageOffset, IO.SeekOrigin.Begin)

```

```

dst.Write(ImgBuff, 0, ImgBuff.Length)

```

```

iconImageOffset += ImgBuff.Length

```

```

Next

```

```

dst.Seek(0, IO.SeekOrigin.Begin)

```

```

IconList.Add(New Icon(dst))

```

```

End Using

```

```

Next

```

```

Return IconList

```

```

End Function

```

```

Public Shared Sub SaveIcon(ByVal Path As String, ByVal mlcon As Icon)

```

```

    Dim fs As IO.FileStream = IO.File.Create(Path)

```

```

    mlcon.Save(fs)

```

```

    fs.Close()

```

```

End Sub

```

```

Private Shared Function EnumResNameProc(ByVal hModule As IntPtr, ByVal lpszType As
IntPtr, ByVal lpszName As IntPtr, ByVal lParam As IntPtr) As Boolean

```

```

    Dim PtrStr As String = Marshal.PtrToStringUni(lpszName)

```

```

    Dim mTyped As New TypedObject

```

```

    If Not IsNothing(PtrStr) Then

```

```

        mTyped.OpType = PtrStr

```

```

    Else

```

```

        mTyped.MainType = lpszName

```

```

    End If

```

```

    PtrToIconList.Add(mTyped)

```

```

    EnumResNameProc = True

```

```

End Function

```

```

End Class

```

Small usage Exemple :

Spoiler [\(Click to View\)](#)

PHP Code

```
<?php
```

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

```
Public Class Form1
```

```
    Sub New()
```

```
        AllowDrop = True
```

```
        ' This call is required by the designer.
```

```
        InitializeComponent()
```

```
        ' Add any initialization after the InitializeComponent() call.
```

```
    End Sub
```

```
    Private Sub CM_CLICK(sender As Object, e As EventArgs)
```

```
        Dim sfd As New SaveFileDialog With {.Filter = "Icon files (*.ico)|*.ico"}
```

```
        DirectCast(sender, ContextMenuStrip).Visible = False
```

```
        If sfd.ShowDialog = Windows.Forms.DialogResult.OK Then
```

```
            Dim mList As List(Of Icon) = DirectCast(DirectCast(sender, ContextMenuStrip).Tag(1), List(
Of Icon))
```

```
            Natives.Savelcon(sfd.FileName, mList(DirectCast(sender, ContextMenuStrip).Tag(0)))
```

```
            MsgBox("Icon Saved !")
```

```
        End If
```

```
    End Sub
```

```
    Private Sub Form1_DragDrop(sender As Object, e As DragEventArgs) Handles Me.DragDrop
```

```
        Dim Data() As String = e.Data.GetData(DataFormats.FileDrop)
```

```
        Dim mList As List(Of Icon) = Natives.GetIcons(Data(0))
```

```
        Dim OldLoc As Integer = 0
```

```
        For I As Integer = 0 To mList.Count - 1
```

```
            Dim pict As New PictureBox()
```

```
            pict.Size = Icon.Size
```

```
            pict.Image = mList(I).ToBitmap()
```

```
            pict.Location = New Point(OldLoc, 0)
```

```
            Dim CM As New ContextMenuStrip
```

```
            pict.ContextMenuStrip = CM
```

```
            CM.Items.Add("Save")
```

```
            CM.Tag = New Object() {I, mList}
```

```
            AddHandler CM.Click, AddressOf CM_CLICK
```

```
            OldLoc += pict.Width + 3
```

```
            Controls.Add(pict)
```

```
        Next
```

```
    End Sub
```

```
    Private Sub Form1_DragEnter(sender As Object, e As DragEventArgs) Handles Me.DragEnter
```

```
        If e.Data.GetDataPresent(DataFormats.FileDrop) Then
```

```
            e.Effect = DragDropEffects.Copy
```

```
        End If
```

```
    End Sub
```

```
End Class
```

TODO:

- clear the code (a bit messy)
- Add BMP to ICO conversion : (As I stated up there, their struct is basically the same)

Enjoy !?

Earn - January 28, 2015, 9:07 pm

Great release. I am sure many people will find this handy. Also, thanks for the link to the MSDN page. That will be a good read.

By the way, the image in the OP is broken because it is not a direct link. I re-uploaded it to Imgur to make things easier.

SubGenius - January 28, 2015, 10:52 pm

Seems cool.

Nice job fransoua la baguette.

AceVoid - January 29, 2015, 4:37 am

Nice share bennz! Find little programs like this very useful.

SIM-1456245 - January 29, 2015, 7:45 am

Very nice that you released this. Looks good.

SirenNinja - January 29, 2015, 9:35 am

I've never seen anything like this before. I mean, I've seen the ones where you can extract a single icon, but not like this. o.O

Dubitus™ - January 29, 2015, 10:00 am

Nice release ibennz, Echo reppin' the VBNET section.

Only3 - January 29, 2015, 10:49 am

theres icon apis u can use to do it easily

<http://www.pinvoke.net/default.aspx/shell32.extracticonex>

<http://www.pinvoke.net/default.aspx/shell32.extracticon>

0xBB - January 29, 2015, 6:55 pm

(Jan 29, 2015 11:49 AM)Only3 Wrote: [\[CITE\]](#) theres icon apis u can use to do it easily

<http://www.pinvoke.net/default.aspx/shell32.extracticonex>

<http://www.pinvoke.net/default.aspx/shell32.extracticon>

You need to know the icon index, if there is small or big. not accurate, does this preserve each layers of the extracted icon?

Quote:phiconLarge [out, optional]

Type: HICON*

An array of icon handles that receives handles to the large icons extracted from the file. If this parameter is NULL, no large icons are extracted from the file.

It seems like it get the larger icon in the dir of icons , so doubt it does. Quality will probably be bad.

iTeDDy - February 18, 2015, 4:47 am

Great release! Thanks man ?

raz3N - February 19, 2015, 11:08 am

Very Nice, great release, thank you for the source code :)

Adderall - *February 19, 2015, 11:25 am*

Nice function in VB.net!

Thank you for sharing the source. ?

Ajax1337 - *February 24, 2015, 9:06 am*

Nice share:) THX!

Greetz! Ajax1337