

Forum:[Hack Forums](#)

Thread: [\[SRC\] \[C#\] Resource Manager](#)

affixiate - December 14, 2010, 8:24 am

The "affixiate" way of managing application resources using WinAPI.

This class allows you to easily read/write to your own applications resource section as well as that of others.

```
Code using System;
using System.Globalization;
using System.Runtime.InteropServices;
```

```
/*
 * Title: CResourceManager.cs
 * Description: Class that wraps reading and writing to resource files using WinAPI.
 *
 * Developed by: affixiate
 * Release date: December 14, 2010
 * Released on: https://hackforums.net
 *
 * Comments: If you use this code, I require you to give me credits. Don't be a ripper! ;]
```

```
public class CResourceManager
{
    private readonly string _file;
    private readonly ushort _currentCultureId;

    /// <summary>
    /// Initializes a new instance of the <see cref="CResourceManager"/> class.
    /// </summary>
    /// <param name="file">The full path of the file to manage resources of.</param>
    public CResourceManager(string file)
    {
        _file = file;
        _currentCultureId = MAKELANGID(PRIMARYLANGID((ushort)
CultureInfo.CurrentCulture.LCID), SUBLANGID((ushort) CultureInfo.CurrentCulture.LCID));
    }

    /// <summary>
    /// Writes to a specified resource.
    /// </summary>
    /// <param name="resourceType">Type of the resource.</param>
    /// <param name="resourceName">Name of the resource.</param>
    /// <param name="bufData">The data to write.</param>
    /// <returns>
    ///     <c>true</c> if operation was successful; otherwise <c>false</c>.
    /// </returns>
```

```

public bool Write<TA, TB>(TA resourceType, TB resourceName, byte[] bufData)
{
    IntPtr hResource = IntPtr.Zero;
    GCHandle gcResource = GCHandle.Alloc(bufData, GCHandleType.Pinned);

    try
    {
        hResource = BeginUpdateResource(_file, false);
        if (hResource == IntPtr.Zero)
            throw new Exception();

        bool ret = false;
        if (typeof(TA) == typeof(int) && typeof(TB) == typeof(int))
            ret = UpdateResource(hResource, (int)(object)resourceType, (int)(object)resourceName,
            _currentCultureId, gcResource.AddrOfPinnedObject(), (uint)bufData.Length);
        else if (typeof(TA) == typeof(int) && typeof(TB) == typeof(string))
            ret = UpdateResource(hResource, (int)(object)resourceType,
            (string)(object)resourceName, _currentCultureId, gcResource.AddrOfPinnedObject(),
            (uint)bufData.Length);
        else if (typeof(TA) == typeof(string) && typeof(TB) == typeof(int))
            ret = UpdateResource(hResource, (string)(object)resourceType,
            (int)(object)resourceName, _currentCultureId, gcResource.AddrOfPinnedObject(),
            (uint)bufData.Length);
        else if (typeof(TA) == typeof(string) && typeof(TB) == typeof(string))
            ret = UpdateResource(hResource, (string)(object)resourceType,
            (string)(object)resourceName, _currentCultureId, gcResource.AddrOfPinnedObject(),
            (uint)bufData.Length);

        if (!ret)
            throw new Exception();
    }
    catch
    {
        return false;
    }
    finally
    {
        if (hResource != IntPtr.Zero)
            EndUpdateResource(hResource, false);

        gcResource.Free();
    }

    return true;
}

/// <summary>
/// Reads the specified resource.
/// </summary>
/// <param name="resourceType">Type of the resource.</param>
/// <param name="resourceName">Name of the resource.</param>
/// <returns>The contents of that resource.</returns>
public byte[] Read<TA, TB>(TA resourceType, TB resourceName)

```

```

{
    IntPtr hModule = IntPtr.Zero;

    try
    {
        hModule = LoadLibraryEx(_file, IntPtr.Zero, 0x2 /* LoadLibraryAsDatafile */);
        if (hModule == IntPtr.Zero)
            throw new Exception();

        IntPtr hResource = IntPtr.Zero;

        if (typeof(TA) == typeof(int) && typeof(TB) == typeof(int))
            hResource = FindResource(hModule, (int)(object)resourceName,
(int)(object)resourceType);
        else if (typeof(TA) == typeof(int) && typeof(TB) == typeof(string))
            hResource = FindResource(hModule, (int)(object)resourceName,
(string)(object)resourceType);
        else if (typeof(TA) == typeof(string) && typeof(TB) == typeof(int))
            hResource = FindResource(hModule, (string)(object)resourceName,
(int)(object)resourceType);
        else if (typeof(TA) == typeof(string) && typeof(TB) == typeof(string))
            hResource = FindResource(hModule, (string)(object)resourceName,
(string)(object)resourceType);

        if (hResource == IntPtr.Zero)
            throw new Exception();

        uint dwResourceSize = SizeofResource(hModule, hResource);
        if (dwResourceSize == 0)
            throw new Exception();

        IntPtr hGlobal = LoadResource(hModule, hResource);
        if (hGlobal == IntPtr.Zero)
            throw new Exception();

        IntPtr hLock = LockResource(hGlobal);
        if (hLock == IntPtr.Zero)
            throw new Exception();

        var buf = new byte[dwResourceSize];
        Marshal.Copy(hLock, buf, 0, buf.Length);
        return buf;
    }
    catch
    {
        return null;
    }
    finally
    {
        if (hModule != IntPtr.Zero)
            FreeLibrary(hModule);
    }
}

```

```

/// <summary>
/// Creates a language identifier from a primary language identifier and a sub-language identifier.
/// </summary>
/// <param name="primaryLanguage">The primary language.</param>
/// <param name="subLanguage">The sub language.</param>
/// <returns>The language identifier.</returns>
private static ushort MAKELANGID(byte primaryLanguage, byte subLanguage)
{
    return (ushort)((subLanguage << 10) | primaryLanguage);
}

```

```

/// <summary>
/// Extracts a primary language identifier from a language identifier.
/// </summary>
/// <param name="languageId">The language id.</param>
/// <returns>The primary language identifier.</returns>
private static byte PRIMARYLANGID(ushort languageId)
{
    return (byte)(languageId & 0x3ff);
}

```

```

/// <summary>
/// Extracts a sublanguage identifier from a language identifier.
/// </summary>
/// <param name="languageId">The language id.</param>
/// <returns>The secondary language identifier.</returns>
private static byte SUBLANGID(ushort languageId)
{
    return (byte)(languageId >> 10);
}

```

#region WinNT Definitions

```

#region WinAPI
[DllImport("kernel32.dll", SetLastError = true)]
private static extern IntPtr LoadLibraryEx(string lpFileName, IntPtr hFile, uint dwFlags);

```

```

[DllImport("kernel32.dll", SetLastError = true)]
private static extern bool FreeLibrary(IntPtr hModule);

```

```

#region FindResource
[DllImport("kernel32.dll", SetLastError = true)]
private static extern IntPtr FindResource(IntPtr hModule, int lpName, int lpType);

```

```

[DllImport("kernel32.dll", SetLastError = true)]
private static extern IntPtr FindResource(IntPtr hModule, int lpName, string lpType);

```

```

[DllImport("kernel32.dll", SetLastError = true)]
private static extern IntPtr FindResource(IntPtr hModule, string lpName, int lpType);

```

```

[DllImport("kernel32.dll", SetLastError = true)]
private static extern IntPtr FindResource(IntPtr hModule, string lpName, string lpType);

```

#endregion

```
[DllImport("kernel32.dll", SetLastError = true)]
private static extern IntPtr LoadResource(IntPtr hModule, IntPtr hResInfo);
```

```
[DllImport("kernel32.dll", SetLastError = true)]
private static extern uint SizeofResource(IntPtr hModule, IntPtr hResInfo);
```

```
[DllImport("kernel32.dll", SetLastError = true)]
private static extern IntPtr LockResource(IntPtr hResData);
```

```
[DllImport("kernel32.dll", SetLastError = true)]
private static extern IntPtr BeginUpdateResource(string pFileName,
[MarshalAs(UnmanagedType.Bool)] bool bDeleteExistingResources);
```

#region UpdateResource

```
[DllImport("kernel32.dll", SetLastError = true)]
private static extern bool UpdateResource(IntPtr hUpdate, int lpType, int lpName, ushort
wLanguage, IntPtr lpData, uint cbData);
```

```
[DllImport("kernel32.dll", SetLastError = true)]
private static extern bool UpdateResource(IntPtr hUpdate, int lpType, string lpName, ushort
wLanguage, IntPtr lpData, uint cbData);
```

```
[DllImport("kernel32.dll", SetLastError = true)]
private static extern bool UpdateResource(IntPtr hUpdate, string lpType, int lpName, ushort
wLanguage, IntPtr lpData, uint cbData);
```

```
[DllImport("kernel32.dll", SetLastError = true)]
private static extern bool UpdateResource(IntPtr hUpdate, string lpType, string lpName, ushort
wLanguage, IntPtr lpData, uint cbData);
#endregion
```

```
[DllImport("kernel32.dll", SetLastError = true)]
private static extern bool EndUpdateResource(IntPtr hUpdate, bool fDiscard);
```

#endregion

#endregion

}

Example usage:

```
Code CResourceManager resourceManager = new
CResourceManager(Application.ExecutablePath);
byte[] buf = resourceManager.Read("BITMAP", "IDB_IMAGE");
```

//...

```
resourceManager.Write("BITMAP", "IDB_IMAGE", buf);
```

If you use this code, it would be most excellent if you could maintain the credits. I'm not asking for cash or beer. This is the least you can do for such high quality work, no? ;)

Don't be a ripper.

/affixiate

P.S. All constructive criticism as well as questions and general comments are welcome.

no1me - December 14, 2010, 2:22 pm

good stuff gone in HF

c# to vb.net

Code Imports System.Globalization

Imports System.Runtime.InteropServices

,

* Title: CResourceManager.cs

* Description: Class that wraps reading and writing to resource files using WinAPI.

*

* Developed by: affixiate

* Release date: December 14, 2010

* Released on: <https://hackforums.net>

*

* Comments: If you use this code, I require you to give me credits. Don't be a ripper! ;]

,

Public Class CResourceManager

Private ReadOnly _file As String

Private ReadOnly _currentCultureId As UShort

''' <summary>

''' Initializes a new instance of the <see cref="CResourceManager"/> class.

''' </summary>

''' <param name="file">The full path of the file to manage resources of.</param>

Public Sub New(file As String)

 _file = file

 _currentCultureId =

 MAKELANGID(PRIMARYLANGID(CUShort(CultureInfo.CurrentCulture.LCID)),

 SUBLANGID(CUShort(CultureInfo.CurrentCulture.LCID)))

End Sub

''' <summary>

''' Writes to a specified resource.

''' </summary>

''' <param name="resourceType">Type of the resource.</param>

''' <param name="resourceName">Name of the resource.</param>

''' <param name="bufData">The data to write.</param>

''' <returns>

''' <c>true</c> if operation was successful; otherwise <c>false</c>.

''' </returns>

Public Function Write(Of TA, TB)(resourceType As TA, resourceName As TB, bufData As Byte())

As Boolean

 Dim hResource As IntPtr = IntPtr.Zero

 Dim gcResource As GCHandle = GCHandle.Alloc(bufData, GCHandleType.Pinned)

Try

```

    hResource = BeginUpdateResource(_file, False)
    If hResource = IntPtr.Zero Then
        Throw New Exception()
    End If

    Dim ret As Boolean = False
    If GetType(TA) Is GetType(Integer) AndAlso GetType(TB) Is GetType(Integer) Then
        ret = UpdateResource(hResource, CInt(DirectCast(resourceType, Object)),
CInt(DirectCast(resourceName, Object)), _currentCultureId, gcResource.AddrOfPinnedObject(),
CUInt(bufData.Length))
    ElseIf GetType(TA) Is GetType(Integer) AndAlso GetType(TB) Is GetType(String) Then
        ret = UpdateResource(hResource, CInt(DirectCast(resourceType, Object)),
DirectCast(DirectCast(resourceName, Object), String), _currentCultureId,
gcResource.AddrOfPinnedObject(), CUInt(bufData.Length))
    ElseIf GetType(TA) Is GetType(String) AndAlso GetType(TB) Is GetType(Integer) Then
        ret = UpdateResource(hResource, DirectCast(DirectCast(resourceType, Object), String),
CInt(DirectCast(resourceName, Object)), _currentCultureId, gcResource.AddrOfPinnedObject(),
CUInt(bufData.Length))
    ElseIf GetType(TA) Is GetType(String) AndAlso GetType(TB) Is GetType(String) Then
        ret = UpdateResource(hResource, DirectCast(DirectCast(resourceType, Object), String),
DirectCast(DirectCast(resourceName, Object), String), _currentCultureId,
gcResource.AddrOfPinnedObject(), CUInt(bufData.Length))
    End If

    If Not ret Then
        Throw New Exception()
    End If
Catch
    Return False
Finally
    If hResource <> IntPtr.Zero Then
        EndUpdateResource(hResource, False)
    End If

    gcResource.Free()
End Try

Return True
End Function

''' <summary>
''' Reads the specified resource.
''' </summary>
''' <param name="resourceType">Type of the resource.</param>
''' <param name="resourceName">Name of the resource.</param>
''' <returns>The contents of that resource.</returns>
Public Function Read(Of TA, TB)(resourceType As TA, resourceName As TB) As Byte()
    Dim hModule As IntPtr = IntPtr.Zero

    Try
        ' LoadLibraryAsDatafile
        hModule = LoadLibraryEx(_file, IntPtr.Zero, &H2)
        If hModule = IntPtr.Zero Then

```

```
    Throw New Exception()  
End If
```

```
Dim hResource As IntPtr = IntPtr.Zero
```

```
    If GetType(TA) Is GetType(Integer) AndAlso GetType(TB) Is GetType(Integer) Then  
        hResource = FindResource(hModule, CInt(DirectCast(resourceName, Object)),  
CInt(DirectCast(resourceType, Object)))  
    ElseIf GetType(TA) Is GetType(Integer) AndAlso GetType(TB) Is GetType(String) Then  
        hResource = FindResource(hModule, CInt(DirectCast(resourceName, Object)),  
DirectCast(DirectCast(resourceType, Object), String))  
    ElseIf GetType(TA) Is GetType(String) AndAlso GetType(TB) Is GetType(Integer) Then  
        hResource = FindResource(hModule, DirectCast(DirectCast(resourceName, Object),  
String), CInt(DirectCast(resourceType, Object)))  
    ElseIf GetType(TA) Is GetType(String) AndAlso GetType(TB) Is GetType(String) Then  
        hResource = FindResource(hModule, DirectCast(DirectCast(resourceName, Object),  
String), DirectCast(DirectCast(resourceType, Object), String))  
    End If
```

```
    If hResource = IntPtr.Zero Then  
        Throw New Exception()  
    End If
```

```
    Dim dwResourceSize As UInteger = SizeofResource(hModule, hResource)  
    If dwResourceSize = 0 Then  
        Throw New Exception()  
    End If
```

```
    Dim hGlobal As IntPtr = LoadResource(hModule, hResource)  
    If hGlobal = IntPtr.Zero Then  
        Throw New Exception()  
    End If
```

```
    Dim hLock As IntPtr = LockResource(hGlobal)  
    If hLock = IntPtr.Zero Then  
        Throw New Exception()  
    End If
```

```
    Dim buf = New Byte(dwResourceSize - 1) {}  
    Marshal.Copy(hLock, buf, 0, buf.Length)  
    Return buf
```

```
Catch
```

```
    Return Nothing
```

```
Finally
```

```
    If hModule <> IntPtr.Zero Then  
        FreeLibrary(hModule)  
    End If
```

```
End Try
```

```
End Function
```

```
''' <summary>
```

```
''' Creates a language identifier from a primary language identifier and a sub-language identifier.
```

```
''' </summary>
```

```

''' <param name="primaryLanguage">The primary language.</param>
''' <param name="subLanguage">The sub language.</param>
''' <returns>The language identifier.</returns>
Private Shared Function MAKELANGID(primaryLanguage As Byte, subLanguage As Byte) As
UShort
    Return CUShort((subLanguage << 10) Or primaryLanguage)
End Function

''' <summary>
''' Extracts a primary language identifier from a language identifier.
''' </summary>
''' <param name="languageId">The language id.</param>
''' <returns>The primary language identifier.</returns>
Private Shared Function PRIMARYLANGID(languageId As UShort) As Byte
    Return CByte(languageId And &H3ff)
End Function

''' <summary>
''' Extracts a sublanguage identifier from a language identifier.
''' </summary>
''' <param name="languageId">The language id.</param>
''' <returns>The secondary language identifier.</returns>
Private Shared Function SUBLANGID(languageId As UShort) As Byte
    Return CByte(languageId >> 10)
End Function

#Region "WinNT Definitions"

#Region "WinAPI"
<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function LoadLibraryEx(lpFileName As String, hFile As IntPtr, dwFlags As
UInteger) As IntPtr
End Function

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function FreeLibrary(hModule As IntPtr) As Boolean
End Function

#Region "FindResource"
<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function FindResource(hModule As IntPtr, lpName As Integer, lpType As
Integer) As IntPtr
End Function

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function FindResource(hModule As IntPtr, lpName As Integer, lpType As String)
As IntPtr
End Function

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function FindResource(hModule As IntPtr, lpName As String, lpType As Integer)
As IntPtr
End Function

```

```

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function FindResource(hModule As IntPtr, lpName As String, lpType As String)
As IntPtr
End Function
#End Region

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function LoadResource(hModule As IntPtr, hResInfo As IntPtr) As IntPtr
End Function

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function SizeofResource(hModule As IntPtr, hResInfo As IntPtr) As UInteger
End Function

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function LockResource(hResData As IntPtr) As IntPtr
End Function

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function BeginUpdateResource(pFileName As String,
<MarshalAs(UnmanagedType.Bool)> bDeleteExistingResources As Boolean) As IntPtr
End Function

#Region "UpdateResource"
<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function UpdateResource(hUpdate As IntPtr, lpType As Integer, lpName As
Integer, wLanguage As UShort, lpData As IntPtr, cbData As UInteger) As Boolean
End Function

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function UpdateResource(hUpdate As IntPtr, lpType As Integer, lpName As
String, wLanguage As UShort, lpData As IntPtr, cbData As UInteger) As Boolean
End Function

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function UpdateResource(hUpdate As IntPtr, lpType As String, lpName As
Integer, wLanguage As UShort, lpData As IntPtr, cbData As UInteger) As Boolean
End Function

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function UpdateResource(hUpdate As IntPtr, lpType As String, lpName As
String, wLanguage As UShort, lpData As IntPtr, cbData As UInteger) As Boolean
End Function
#End Region

<DllImport("kernel32.dll", SetLastError := True)> _
Private Shared Function EndUpdateResource(hUpdate As IntPtr, fDiscard As Boolean) As
Boolean
End Function

#End Region

```

#End Region

```
End ClassDim resourceManager As New CResourceManager(Application.ExecutablePath)
Dim buf As Byte() = resourceManager.Read("BITMAP", "IDB_IMAGE")

resourceManager.Write("BITMAP", "IDB_IMAGE", buf)
tnx
```

Grieve™ - December 14, 2010, 2:34 pm

nice work again from you affixiate. your a great new memeber. keep it up!

affixiate - December 14, 2010, 6:47 pm

(Dec 14, 2010 03:34 PM)Grieve™ Wrote: [\[CITE\]](#) nice work again from you affixiate. your a great new memeber. keep it up!
tyvm sir.

I'll be sure to contribute more ?.

Mr. Anderson - December 14, 2010, 7:09 pm

I am laughing so hard at the all the thread views and only 3 comments? You should encrypt your sources so that they cannot be leached so easily.

Nice job by the way :)

pr0t0typ3 - December 14, 2010, 9:52 pm

Nice job keep posting put I know you got many good ideas.

sh@rp - November 16, 2012, 7:23 am

Hi,

Anyone know how to "Replace" a resource because when I use this function it doesn't replace the existing resource but it created another one in the treeview with the same name, the same type and the same language... ??

[img]<http://imabox.fr/u20/1216111631s470mm15.png>[/img]

I tried lot of example on Internet by using Win API but I'm facing the same problem (Add and don't update or replace It !) !

I know ResHack and other tools in command line to do that stuff but I want to find a solution first. ;

This is actually my little snippet (Not optimize I know !) :

```
Code public class Cls_UpdateResource
{
```

```
    [DllImport("kernel32.dll", SetLastError = true)]
    private static extern IntPtr BeginUpdateResource(string pFileName,
    [MarshalAs(UnmanagedType.Bool)]
    bool bDeleteExistingResources);
```

```
    [DllImport("kernel32.dll", SetLastError = true)]
    private static extern bool UpdateResource(IntPtr hUpdate, string lpType, string lpName, ushort
```

```
wLanguage, IntPtr lpData, uint cbData);
```

```
[DllImport("kernel32.dll", SetLastError = true)]
```

```
private static extern bool EndUpdateResource(IntPtr hUpdate, bool fDiscard);
```

```
public static bool UpdateFileResource(string executablefilepath, string resourceFilePath)
```

```
{
```

```
    IntPtr updateHandle = BeginUpdateResource(executablefilepath, false);
```

```
    if (updateHandle == IntPtr.Zero) {
```

```
        return false;
```

```
    }
```

```
    System.IO.Stream resourcestream = File.OpenRead(resourceFilePath);
```

```
    byte[] buffer = new byte[resourcestream.Length];
```

```
    resourcestream.Read(buffer, 0, Convert.ToInt32(buffer.Length));
```

```
    if (!UpdateResource(updateHandle, "UIFILE", "23", Convert.ToUInt16(1033),
```

```
Marshal.UnsafeAddrOfPinnedArrayElement(buffer, 0), buffer.Length)) {
```

```
        return false;
```

```
    }
```

```
    if (!EndUpdateResource(updateHandle, false)) {
```

```
        return false;
```

```
    }
```

```
    resourcestream.Close();
```

```
    return true;
```

```
}
```

```
}
```

I tried this but no result too :

```
Code if (!UpdateResource(updateHandle, "UIFILE", "23", Convert.ToUInt16(1033), IntPtr.Zero, 0)) {
```

```
    return false;
```

```
}
```

Thx