MATEC Web of Conferences **44**, 01087 (2016)
DOI: 10.1051/matecconf/20164401087

Invoking Device Driver from .NET Managed Code

Peng XIE [1,a], Li AN [1] and Hong Mei ZHANG[1]

[1]*College of Science, Air Force Engineering University, Xi'an, China*

Abstract. It's a real problem to visit a low-level device driver from a high-level application based on the .NET platform managed code in software developing practice .This paper concerns how to solve it by a runtime interoperation services provided by Microsoft .NET Common Language Runtime（CLR） . A real world example is given in the end .

1. INTRODUCTION

The .NET is a set of software circumstance used for development and deployment of enterprise application, which introduced by Microsoft for fight with Java. we also realized .NET deeply in the process of developing a information system. But it is a fact that .NET is inferior to unmanaged native code on visiting low-level system. However, not all enterprise applications are running on high-lever virtual machine instead of low-level system. Whether .NET can visit low-level device or not, and how to vist, that is a problem we encountered in developing a information system, the resolving methods as follows:

The system we developed is a typical high-level manage application. However ,if realized to show clients' information base on telephone voice box in the Customer Relationship Management system (CRM), it requires to visit device driver provided by virtual machine, so as to set in the way of hardware work models, and extract the Caller's ID when the phone rang. How to complete it? After queried VS and MSDN, we found that use the runtime interoperation services provided by .NET Common Language Runtime (CLR) with that the .NET managed code can realize unmanaged function in visiting Dynamic Linking Library (DLL), which for Microsoft could visit Windows API under .NET circumstance.

2. VIST UNMANAGED CODE SYSTEM THROUGH .NET RUNTIME INTEROPERATION SERVICE

The runtime interoperation service provided by the .NET will contribute to look for management Code. In detail, when unmanaged function is operated by runtime interoperation services, the performing operations as following steps:

To Look for DLL including the function.

To load the DLL into memory

To locate the address of function in memory and pushes its arguments onto the stack, marshling date as required.

To transfer the control right to unmanaged function

The runtime interoperation services will introduce for managed invoking caller abnormal that generated by unmanaged function

[a] Corresponding author: xpf68@163.com

As Figure 1:

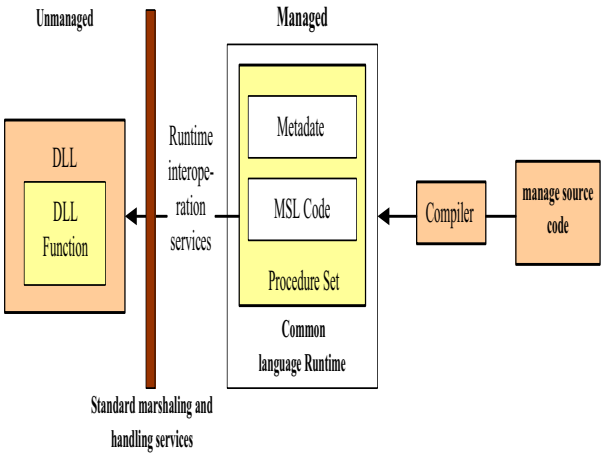

Figure 1.unmanaged and managed

3. THE METHOD OF C# VISITING THE FUNCTION FROM UNMANAGED DLL

a. Identity function of DLL

DLL function identification contains the name or serial number of function, as well as realizes the DLL file's name. For example: Specify MessageBox function in User32.dll, we need identify it (the MessageBox function)and the position (User32.dll, User 32 or user 32.)

If not specified, all character set showed by Charset field will default to ANSI.

The programmer could rename the unmanaged function in code to any required name, which need to map to DLL initial entry point.

b. Create managed-class for holding DLL function.

In order to encapsulate, it always packs the DLL function into managed-clas . Although not always do like this in every situation, the providing managed-class is very convenient method. The programmer could:

To declare DLL function in existing class.

To create a class for every DLL function, in case to find easily by isolating function from each other.

To create a class for a group of related DLL function, in case to form logical grouping and reduce system overhead.

c . Create DLL function prototype in managed-class.

In this situation, it is need to define the static method for every DLL function in managed-class, and pay special attention to substitute unmanaged date types to managed date types. The Corresponding relationship between date types in C function and Wtypes.h definition and unmanaged date types as Table 1:

Table 1. definition and unmanaged date types.

Unmanaged type of Wtypes.h	Unmanaged C Language type	Managed-class Name	C# date type
HANDLE	void*	System.IntPtr	No internal
BYTE	unsigned char	System.Byte	byte
SHORT	short	System.Int16	short
WORD	unsigned short	System.UInt16	ushort
INT	int	System.Int32	Integer
UINT	unsigned int	System.UInt32	uint
LONG	long	System.Int32	Integer
BOOL	long	System.Int32	Integer
DWORD	unsigned long	System.UInt32	uint
ULONG	unsigned long	System.UInt32	uint
CHAR	char	System.Char	char
LPSTR	char*	System.String or System.StringBuilder	String or StringBuil-der
LPCSTR	Const char*	System.String or System.String-	String or String-

		Builder	Builder
LPWSTR	wchar_t*	System.String or System.StringBuil-der	String or StringBuil-der
LPCWSTRF	Const wchar_t*	System.String or System.StringBuil-der	String or StringBuil-der
FLOAT	Float	System.Single	float
DOUBLE	Double	System.Double	double

d. Set property field

The managed prototype declaration in function can show setting property field, so as to define the behavior of managed code. For example, the character sets and calling conventions used in passing method arguments. There are related property fields and defaults related with platform runtime interoperation services in table 2:

Table2. Fields and Explanation

Fields	Explanation
BestFitMapping	To switch the most appropriate mapping
EntryPoint	To assign the invoking DLL entry point
ExactSpelling	To modify entries to maintain corresponding with character set. Different programming language, different default.
PreserveSig	To control managed method signature : switch to back HRESULT or not,s with unmanaged signature of additional parameters[out, retval]. The default is true (should not transform signature)
SetLastError	To make invoking party confirm there was an error or not in using Marshal.GetLastWin32Error API function.

e. Invoking DLL Function

4. VISIT TELEPHONE VOICE BOX DRIVER FORM C#

According to method above, we realized visiting of telephone box driver. The example as following:

a. The C++ invoking interface provided by SDK

```
bool WINAPI SW_Init();
bool WINAPI SW_Free();
void WINAPI SW_SetType(int DeviceType);
int WINAPI SW_GetCount();
int WINAPI SW_GetSN(int nIndex);
int WINAPI SW_ReceiveCID(int nIndex, char *lpBuf);
```

b. The C# packing to meet requirement of platform runtime services

```
using System.Runtime.InteropServices;
public class CCIDFun
{       [DllImport("SWind.dll", EntryPoint="SW_Init")]
public static extern int SW_Init();
[DllImport("SWind.dll", EntryPoint="SW_Free")]
public static extern int SW_Free();
[DllImport("SWind.dll", EntryPoint="SW_SetType")]
public static extern void SW_SetType(int DeviceType);
[DllImport("SWind.dll", EntryPoint="SW_GetCount")]
public static extern int SW_GetCount();
[DllImport("SWind.dll", EntryPoint="SW_GetSN")]
public static extern int SW_GetSN(int nIndex);
[DllImport("SWind.dll", EntryPoint="SW_ReceiveCID")]
public unsafe static extern int SW_ReceiveCID(int nIndex, StringBuilder lpBuf);
}
```

c. Test code

```
……
CCIDFun.SW_Init();
```

```
CCIDFun.SW_SetType(2);
int lineCount = CCIDFun.SW_GetCount();
if ( lineCount!=2 )
MessageBox.Show("Initialization failed!","Message",MessageBoxButtons.OK ,MessageBoxIcon.Information);
else
MessageBox.Show("Initialsuccess!lineCount==2","Message",MessageBoxButtons.OK ,MessageBoxIcon.Information);
……
```

5. CONCLUSION

The direct way is completed in the Visual C++ NET for the mixed programming of managed code and unmanaged code. However, the runtime interoperation service is appropriate in the case of plenty of manage code occasionally invoke unmanaged code, which is proved in practice.

REFERENCES

1. Charles Petzold ． Microsoft C# Windows Programming (Part I; Party II)． Beijing：Peking University Press(2012)
2. Jeremy Greenwood, Boyan Jovanovic. Financial Development , Growth, and The Distribution of Income[J]. Journal Economy,2013,98:1076-1107.
3. Oded Golor, Joseph Zeira. Income Distribution and Macroeconomics[J]. The Reviews of Economic Sduies,2012, 60(1):35-52.
4. Clark, George, Lixin Colin Xu, Heng Zou. Finance and Income in Inequality: Test of Alternative Theories[C]. 2013, NO.2984.

AUTHOR: Peng XIE, Male, Born in Zhumadian City, Henan Province. Master degree, University Lecturer. Electronic science teaching and research department of science college, Air Force Engineering University.
Research Fields: Network and date integration.