How to Write Managed Code That Uses the Mobile Broadband API

Guidelines for Developers of Managed Code Applications

July 10, 2009

Abstract

This document describes how to write applications in managed code that call the mobile broadband API in Windows® 7. It provides guidelines for original equipment manufacturers (OEMs) and independent software vendors (ISVs) for calling the mobile broadband API from managed code through the COM interoperability interface.

This document provides guidelines and key steps for how to develop third-party mobile broadband applications in managed code for Windows 7. It describes the techniques and methods that are used to call the mobile broadband API from managed code applications.

This information applies to the Windows 7 operating system.

References and resources discussed here are listed at the end of this document.

The current version of this paper is maintained on the Web at:  
 <http://www.microsoft.com/whdc/connect/wireless/MBCMgr_W7OS.mspx>

Disclaimer: This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.

© 2009 Microsoft Corporation. All rights reserved.

Microsoft, Visual Studio, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| July 10, 2009 | First publication | | | |

Contents

[Introduction 3](#_Toc234816919)

[Overview 3](#_Toc234816920)

[Installation Requirements 4](#_Toc234816921)

[Calling the Mobile Broadband API from Managed Code 4](#_Toc234816922)

[Generating a COM Interop Assembly 4](#_Toc234816923)

[Using the Type Library Importer to Create a COM Interop Assembly 5](#_Toc234816924)

[Using Visual Studio to Create a COM Interop Assembly 6](#_Toc234816925)

[Instantiating the Mobile Broadband API Manager Interfaces 6](#_Toc234816926)

[Obtaining the Mobile Broadband API Functional Objects 7](#_Toc234816927)

[Obtaining the Mobile Broadband API Interface Objects 8](#_Toc234816928)

[Mobile Broadband Operations and Sample Code 8](#_Toc234816929)

[Error Handling 10](#_Toc234816930)

[Resources 12](#_Toc234816931)

[Appendix: C# Signature for the Mobile Broadband API 13](#_Toc234816932)

# Introduction

This paper provides guidelines and key steps for software developers on how to call the Windows® mobile broadband API from applications that they write in managed code.

The audience for this paper is software developers who have a good understanding of the following:

* How to develop applications in managed code.
* The Windows Component Object Model (COM).
* The C# programming language.
* Interoperating with unmanaged code.

In addition, the audience should be familiar with the Windows mobile broadband API and have a good understanding of the following:

* The mobile broadband API specification.

This specification is an authoritative reference for the interfaces, structures, enumerations, and methods that the mobile broadband API supports.

For more information about the mobile broadband API specification, refer to “[Resources](#_Resources)” later in this paper.

* The sample code for using the mobile broadband API.

This sample code shows how to use this API from unmanaged code. However, the approach for using the API from managed code is basically the same as from unmanaged code. This paper describes those areas where the approach is different for managed code.

For more information about the mobile broadband API sample code, refer to “[Resources](#_Resources)” later in this paper.

# Overview

The Windows mobile broadband API is used to implement connectivity to cellular networks. Third-party application can use this API to control and manage mobile broadband interfaces and connections.

Windows 7 supports the mobile broadband API.

The programming interface of the mobile broadband API is COM-based and developed as unmanaged code. However, developers can use the COM interoperability interface to call the API from managed code.

This paper includes the following sections that discuss the various techniques for calling the mobile broadband API from managed code:

* [Installation Requirements](#_Installation_Requirements)

This section discusses the software components that must be installed in both the development and deployment environments.

* [Calling the Mobile Broadband API from Managed Code](#_Calling_the_Mobile)

This section discusses the required steps to write managed code that calls the mobile broadband API.

* [Resources](#_Resources)

This section provides a list of documents that have additional information.

# Installation Requirements

To develop managed and unmanaged code that calls the Windows mobile broadband API, you must download and install the following components:

* The Windows Software Development Kit (SDK) for Windows 7 and the .NET Framework 3.5 Service Pack1, which must be installed in the development environment. You can install this software in systems that run Windows XP and later versions of Windows.
* The Windows 7 operating system (RC or later versions), which must be installed in the deployment environment.

# Calling the Mobile Broadband API from Managed Code

Because the Windows mobile broadband API consists of unmanaged COM objects and code, you must do the following to call the API from your managed code:

* [Generate a COM interoperability (*interop*) assembly](#_Generating_a_COM).
* [Instantiate the mobile broadband API manager objects](#_Instantiate_the_Mobile).
* [Obtain the mobile broadband API functional objects](#_Obtain_the_Mobile).
* [Handle errors](#_Error_Handling) that are returned from calls to the mobile broadband API.

## Generating a COM Interop Assembly

COM interop assemblies are a bridge between managed and unmanaged code, and map unmanaged COM object members to equivalent managed members.

The .NET common language runtime (CLR) of managed code requires metadata for all data types, which includes COM data types. This metadata is contained in an interop assembly.

You can produce a COM interop assembly by using the following tools:

* The Type Library Importer (*Tlbimp.exe*)
* Microsoft® Visual Studio®

### Using the Type Library Importer to Create a COM Interop Assembly

To create a COM interop assembly by using the Type Library Importer, enter the following code from a command line:

**tlbimp /machine:***MachineType* *TypeLibName*

The command-line parameters are as follows:

* *MachineType*

This specifies the target CPU platform for the COM interop assembly. The following are the valid values for this parameter:

* X86
* X64
* Itanium (IA64)
* *TypeLibName*

This parameter specifies the name of the type library (*.tlb*) file.

By default, the Type Library Importer produces a COM interop assembly that is named *TypeLibName.dll.*

The name of the mobile broadband API type library file is *Mbnapi.tlb*. Versions of this file in the SDK are specific to a CPU platform. In the SDK installation directory, which is *Program Files\Microsoft SDKs* by default, the *Mbnapi.tlb* files are located in the following subdirectories:

* The x86 version of *Mbnapi.tlb* is stored in *Windows\v7.0\Lib*.
* The x64 version of *Mbnapi.tlb* is stored in *Windows\v7.0\Lib\x64*.
* The IA64 version of *Mbnapi.tlb* is stored in *Windows\v7.0\Lib\IA64*.

You should create a COM interop assembly from the CPU-specific version of the *Mbnapi.tlb* file that is appropriate for the deployment environment of your application.

For example, if your application is targeted for only x64 CPU platforms, you must create the COM interop assembly in the following way:

* Copy the *Mbnapi.tlb* file from the SDK’s *Windows\v7.0\Lib\x64 directory* to your development directory.
* Enter the following code from a command line:

tlbimp /machine:x64 Mbnapi.tlb

The COM interop assembly that is created from the *Mbnapi.tlb* file is named *Mbnapi.dll*. You must use this file to reference the managed types of the mobile broadband API during the compilation of managed code application.

**Note:** If the application is built as a 32-bit binary, it can run on both x86 and x64 CPU platforms. On x64 and IA64 CPU platforms, 32-bit applications can run seamlessly because of the support of Windows 32-bit on Windows 64-bit (WOW64) 32-bit emulator. In this case, you must use the x32 version of the *Mbnapi.tlb* file to create the COM interop assembly.

### Using Visual Studio to Create a COM Interop Assembly

If you use Visual Studio to build your managed code application, Visual Studio automatically creates the COM interop assembly.

You must first register the *.tlb* file first before you add a reference to it in a Visual Studio project. Otherwise, you might see the following error when you try to add the reference:

“A reference to mbnapi.tlb could not be added. Please make sure that the file can be accessed, and that it is a valid assembly or COM component.”

If your development environment is a computer that is not running Windows 7, you can register the Mnnapi.tlb library by following these steps:

1. Open a command prompt and move to the SDK’s Windows\v7.0\Lib directory by using the cd command.

2. From the command line, enter the following command:

Regtlibv12 Mbnapi.tlb

**Note:** You are not required to register the .tlb file if your development environment is a computer that is running Windows 7 because the operating system has already registered this file.

The following steps show how to use Visual Studio to reference the mobile broadband API type library and create the COM interop assembly:

1. Open Visual Studio and create a solution. This example uses the name *MyProject* for the project*.* Visual Studio creates a project solution that is named *MyProject.sln.*

2. Add a reference to the *Mbnapi.tlb* file. To do this, follow these steps:

a. Click the **Project** item in the toolbar, and then click **Add Reference** from the drop-down menu.

b. From the **Add Reference** page, click the **Browse** tab.

c. Browse to the *Mbnapi.tlb* file in the SDK’s *Windows\v7.0\Lib* and select it. Then, click **OK**.

As soon as you have added the reference to the *Mbnapi.tlb* file in your project, Visual Studio creates the COM interop assembly.

**Important:** If you want to build your managed code application to run on both x32, x64, and IA64 CPU platforms, you should not compile your application by using the **/platform anycpu** setting in Visual Studio. Although this setting does compile the managed code application in a platform-agnostic mode, the application depends on the COM interop assembly that is platform-specific.

## Instantiating the Mobile Broadband API Manager Interfaces

Three manager interfaces in the mobile broadband API can be instantiated directly. These manager interfaces provide methods for enumerating objects. They also let the application receive notifications for arrival and removal of objects.

The mobile broadband API manager interfaces are as follows:

* **IMbnConnectionManager**, which is also implemented as a COM co-class object that is named **MbnConnectionManager**.
* **IMbnInterfaceManager**, which is also implemented as a COM co-class object that is named **MbnInterfaceManager**.
* **IMbnConnectionProfileManager**, which is also implemented as a COM co-class object that is named **MbnConnectionProfileManager**.

For more information about these and other mobile broadband API manager interfaces, refer to the “Mobile Broadband API Reference” that is listed in “[Resources](#_Resources)” later in this document.

The following code example shows how to instantiate a mobile broadband API manager interface:

Mbn*xxx*Manager mbn*xxx*Mgr = new Mbn*xxx*Manager();

IMbn*xxx*Manager *xxx*Mgr = (IMbn*xxx*Manager) mbn*xxx*Mgr;

**Note:** None of the other mobile broadband API interface objects can be instantiated directly. Your application must first instantiate the manager objects and then use the methods that the manager objects support to enumerate the required interface objects.

## Obtaining the Mobile Broadband API Functional Objects

Every mobile broadband API manager interfaces supports a functional object. You can obtain these objects from the manager interface through the interface’s enumeration method.

The mobile broadband API functional objects are as follows:

* **IMbnConnection**
* **IMbnInterface**
* **IMbnConnectionProfile**

For more information about mobile broadband API functional objects, refer to the “Mobile Broadband API Reference” that is listed in “[Resources](#_Resources)” later in this document.

You can obtain all these functional objects in the same manner. For example, the following code example shows how to obtain the **IMbnInterface** functional object by using the enumeration method of the **IMbnConnectionManager** interface:

MbnConnectionManager mbnConnectionMgr =

new MbnConnectionManager();

IMbnConnectionManager connMgr =

(IMbnConnectionManager) mbnConnectionMgr;

IMbnConnection[ ] arrConn =

(IMbnConnection[ ])connMgr.GetConnections();

## Obtaining the Mobile Broadband API Interface Objects

In unmanaged code, you can obtain various mobile broadband API interface objects by calling the **IUnknown::QueryInterface** method of the **IMbnConnection**, **IMbnInterface**, and **IMbnConnectionProfile** objects.

We highly recommend that you do not call the **IUnknown::QueryInterface** method in your managed code. Instead, you can obtain the interface object in managed code by typecasting the corresponding interface object from the related functional object.

For example, you can use the **IMbnInterface** functional object to obtain the **IMbnRadio** interface. The following code example shows how to obtain this interface through typecasting:

//QI IMbnRadio from IMbnInterface

IMbnRadio radio;

radio = (IMbnRadio) inf;

## Mobile Broadband Operations and Sample Code

You can perform three kinds of mobile broadband operations by using the interface objects from the mobile broadband API:

* Registering for event notifications, such as notification of a change in state of the radio on a mobile broadband device.
* Asynchronous operations.
* Synchronous operations.

The following example shows how to register for mobile broadband notifications through managed code:

//register for notifications

//class to keep global variables

Class Global

{

public static IMbnRadio g\_IMbnRadio = null;

}

//implement the required IMbnXXXEvents

class RadioEventsSink : IMbnRadioEvents

{

public RadioEventsSink () { }

public void OnRadioStateChange (IMbnRadio newInterface)

{

<code>

}

public void OnSetSoftwareRadioStateComplete(IMbnRadio newInterface,uint requestID, int Status)

{

Global.g\_IMbnRadio = newInterface;

<code>

}

}

//instantiate corresponding manager object

MbnInterfaceManager mbnInfMgr = new MbnInterfaceManager();

//1.Get an IConnectionPointContainer interface by typecasting //corresponding manager object.

IConnectionPointContainer icpc =

(IConnectionPointContainer) mbnInfMgr;

//2.Call FindConnectionPoint on the returned interface and pass //corresponding IID\_IMbnXXXEvents to riid say //IID\_IMbnRadioEvents.

Guid IID\_IMbnRadioEvents = typeof(IMbnRadioEvents).GUID;

IConnectionPoint icp;

icpc.FindConnectionPoint(ref IID\_IMbnRadioEvents, out icp);

//3.Call Advise on the returned connection point and pass

//object that implements IMbnXXXEvents.

RadioEventsSink radioEvtsSink = new RadioEventsSink();

icp.Advise(radioEvtsSink, out cookie);

The following example shows how to perform asynchronous mobile broadband operationss through managed code:

// asynchronous operation

MbnInterfaceManager mbnInfMgr =

new MbnInterfaceManager();

IMbnInterfaceManager infMgr =

(IMbnInterfaceManager) mbnInfMgr;

string interfaceID = “{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}”;

try

{

//obtain the IMbnInterface passing interfaceID

IMbnInterface inf= infMgr.GetInterface(interfaceID);

}

catch(Exception e)

{

Console.WriteLine(e.Message);

throw e;

}

//QI IMbnRadio from IMbnInterface

IMbnRadio radio;

radio = (IMbnRadio) inf;

MBN\_RADIO softwareRadioState = MBN\_RADIO.MBN\_RADIO\_ON;

uint requestID = 0;

try

{

//call SetSoftwareRadioState to set the software radio state

radio.SetSoftwareRadioState(softwareRadioState, out requestID);

Console.WriteLine(“Request for SetSoftwareRadioState is submitted successfully with requestID = “ + requestID);

}

catch(Exception e)

{

Console.WriteLine(e.Message);

throw e;

}

//wait for notifications to arrive

//discard the older radio object and use the newer IMbnRadio //interface obatined in the notification //OnSetSoftwareRadioStateComplete

radio = Global.g\_IMbnRadio;

The following example shows how to perform synchronous mobile broadband operations through managed code:

// synchronous operation

MbnConnectionManager mbnConnMgr =

new MbnConnectionManager();

IMbnConnectionManager connMgr =

(IMbnConnectionManager) mbnConnMgr;

try

{

// enumerate the connections using GetConnections

IMbnConnection[ ] arrConn =

(IMbnConnection[ ])connMgr.GetConnections();

}

catch(Exception e)

{

Console.WriteLine(e.Message);

throw e;

}

foreach(IMbnConnection conn in arrConn)

{

Console.WriteLine("Connection ID = " + conn.ConnectionID);

}

## Error Handling

The functions that the mobile broadband API COM interfaces define return a HRESULT error code. These error codes are converted into corresponding C# exceptions. The mobile broadband API error codes that cannot be mapped to generalized C# exceptions are returned as “Exception from HRESULT <0xXXXXXXXX>”.

For synchronous mobile broadband operations, errors are returned from the call to the mobile broadband APIs. You can use the following code sample to catch and handle the errors that the mobile broadband API returns synchronously:

try

{

<code>

}

catch(ArgumentException e)

{

}

catch(NotSupportedException e)

{

}

.

.

.

catch(Exception e)

{

    If (e.Message.Contains(“0x80548201”) {

    }

    else

    if (e.Message.Contains(“0x80548202”) {

    }

.

.

.

}

The error codes in this example, in addition to all mobile broadband API errors, are defined in *Winerror.h*. In the SDK installation directory, which is *Program Files\Microsoft SDKs* by default, this file is located in *Windows\v7.0\Include.*

The callback methods of the mobile broadband API that return error codes do so asynchronously through the method’s *status* parameter. For example, the **OnSetSoftwareRadioStateComplete** method asynchronously returns the error code in the *status parameter*, as follows:

HRESULT OnSetSoftwareRadioStateComplete(

[in] IMbnRadio \*newInterface,

[in] ULONG requestID,

[in] HRESULT status

);

For those mobile broadband API callback methods that return error codes asynchronously, software developers should always check the value of the *status* parameter. A *status value of* nonzero indicates that an error condition has occurred and that the interface objects that the method returns must not be used.

You can also raise the exception mapping to the error code that are returned in the *status* parameter if the value of *Status* is nonzero.

# Resources

For the latest information about the Microsoft Windows family, see the [Windows Web site](http://www.microsoft.com/windows) at <http://www.microsoft.com/windows>.

#### Windows Hardware Developer Central (WHDC):

Home page

<http://www.microsoft.com/whdc/default.mspx>

Wireless Mobile Broadband

<http://www.microsoft.com/whdc/connect/MB/default.mspx>

Windows 7: Mobile Broadband APIs for Application Development  
(WinHEC 2008 presentation)

<http://download.microsoft.com/download/5/E/6/5E66B27B-988B-4F50-AF3A-C2FF1E62180F/MBL-T603_WH08.pptx>

#### MSDN:

Mobile Broadband Home Page

<http://www.microsoft.com/mobilebroadband>

Type Library Importer (*Tlbimp.exe*)

<http://msdn.microsoft.com/en-us/library/tt0cf3sx(VS.80).aspx>

#### Windows Software Development Kit:

Mobile Broadband API Reference

<http://msdn.microsoft.com/en-us/library/dd323269(VS.85).aspx>

Interoperating with Unmanaged Code

<http://msdn.microsoft.com/en-us/library/sd10k43k(VS.80).aspx>

Windows SDK for Windows 7 and .NET Framework 3.5 SP1: RC

<http://www.microsoft.com/downloads/details.aspx?FamilyID=6db1f17f-5f1e-4e54-a331-c32285cdde0c&displaylang=en>

Sample code for using the mobile broadband API

A sample program that uses the mobile broadband API is in the Windows 7 SDK in the *Windows/v7.0/Samples/NetDs/MB/mbapi* subdirectory.

# Appendix: C# Signature for the Mobile Broadband API

For the description and behavior of the available methods, refer to the mobile broadband API specification.

IMbnConnection Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **Connect** | **public Connect (**  **MBN\_CONNECTION\_MODE** *connectionMode*,  **string** *strProfile*,  **out uint** *requestID*  **);** |
| **ConnectionID** | **public string ConnectionID(***get*;**)** |
| **Disconnect** | **public Disconnect (**  **out uint** *requestID*  **);** |
| **GetActivationNetworkError** | **public uint GetActivationNetworkError()** |
| **GetConnectionState** | **public void GetConnectionState(**  **out MBN\_ACTIVATION\_STATE** *connectionState***,**  **out string** *profileName*  **);** |
| **GetVoiceCallState** | **public MBN\_VOICE\_CALL\_STATE GetVoiceCallState();** |
| **InterfaceID** | **public string InterfaceID(***get*;**);** |

IMbnConnectionContext Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **GetProvisionedContexts** | **public System.Array GetProvisionedContexts();** |
| **SetProvisionedContext** | **public SetProvisionedContext (**  **MBN\_CONTEXT** *provisionedContext***,**  **string** *providerID***,**  **out uint** *requestID*  **);** |

IMbnConnectionContextEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnProvisionedContextListChange** | **public OnProvisionedContextListChange(**  **IMbnConnectionContext** *newInterface*  **);** |
| **OnSetProvisionedContextComplete** | **public OnSetProvisionedContextComplete (**  **IMbnConnectionContext** *newInterface***,**  **uint** *requestID***,**  **int** *status*  **);** |

IMbnConnectionEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnConnectComplete** | **public OnConnectComplete (**  **IMbnConnection** *newConnection***,**  **uint** *requestID***,**  **int** *status*  **);** |
| **OnConnectStateChange** | **public OnConnectStateChange(**  **IMbnConnection** *newConnection*  **);** |
| **OnDisconnectComplete** | **public OnDisconnectComplete(**  **IMbnConnection** *newConnection***,**  **uint** *requestID***,**  **int** *status*  **);** |
| **OnVoiceCallStateChange** | **public OnVoiceCallStateChange(**  **IMbnConnection** *newConnection*  **);** |

IMbnConnectionManager Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **GetConnection** | **public IMbnConnection GetConnection (**  **string** *connectionID*  **);** |
| **GetConnections** | **public System.Array GetConnections();** |

IMbnConnectionManagerEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnConnectionArrival** | **public void OnConnectionArrival(**  **IMbnConnection** *newConnection*  **);** |
| **OnConnectionRemoval** | **public void OnConnectionRemoval(**  **IMbnConnection** *oldConnection*  **);** |

IMbnConnectionProfile Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **Delete** | **public void Delete();** |
| **GetProfileXmlData** | **public string GetProfileXmlData();** |
| **UpdateProfile** | **public void UpdateProfile(**  **string** *strProfile*  **);** |

IMbnConnectionProfileEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnProfileUpdate** | **public OnProfileUpdate(**  **IMbnConnectionProfile** *newProfile*  **);** |

IMbnConnectionProfileManager Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **CreateConnectionProfile** | **public void CreateConnectionProfile(**  **string** *xmlProfile*  **);** |
| **GetConnectionProfile** | **public**  **IMbnConnectionProfile GetConnectionProfile(**  **IMbnInterface** *mbnInterface***,**  **string** *profileName*  **);** |
| **GetConnectionProfiles** | **public System.Array GetConnectionProfiles(**  **IMbnInterface** *mbnInterface*  **);** |

IMbnConnectionProfileManagerEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnConnectionProfileArrival** | **public void OnConnectionProfileArrival(**  **IMbnConnectionProfile** *newConnectionProfile*  **);** |
| **OnConnectionProfileRemoval** | **public void OnConnectionProfileRemoval(**  **IMbnConnectionProfile** *oldConnectionProfile*  **);** |

IMbnInterface Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **GetConnection** | **public IMbnConnection GetConnection ();** |
| **GetHomeProvider** | **public MBN\_PROVIDER GetHomeProvider();** |
| **GetInterfaceCapability** | **public**  **MBN\_INTERFACE\_CAPS GetInterfaceCapability();** |
| **GetPreferredProviders** | **public System.Array GetPreferredProviders();** |
| **GetReadyState** | **public MBN\_READY\_STATE GetReadyState();** |
| **GetSubscriberInformation** | **public**  **IMbnSubscriberInformation GetSubscriberInformation();** |
| **GetVisibleProviders** | **public System.Array GetVisibleProviders(**  **out uint** *age*  **);** |
| **InEmergencyMode** | **public bool InEmergencyMode();** |
| **InterfaceID** | **public string InterfaceID(***get***;);** |
| **ScanNetwork** | **public void ScanNetwork(**  **out uint** *requestID*  **);** |
| **SetPreferredProviders** | **public SetPreferredProviders(**  **System.Array** *PreferredProviders***,**  **out uint** *requestID*  **);** |

IMbnInterfaceEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnEmergencyModeChange** | **public void OnEmergencyModeChange(**  **IMbnInterface** *newInterface*  **);** |
| **OnHomeProviderAvailable** | **public void OnHomeProviderAvailable(**  **IMbnInterface** *newInterface*  **);** |
| **OnInterfaceCapabilityAvailable** | **public void OnInterfaceCapabilityAvailable(**  **IMbnInterface** *newInterface*  **);** |
| **OnPreferredProvidersChange** | **public void OnPreferredProvidersChange(**  **IMbnInterface** *newInterface*  **);** |
| **OnReadyStateChange** | **public void OnReadyStateChange(**  **IMbnInterface** *newInterface*  **);** |
| **OnScanNetworkComplete** | **public void OnScanNetworkComplete(**  **IMbnInterface** *newInterface***,**  **uint** *requestID***,**  **uint** *status*  **);** |
| **OnSetPreferredProvidersComplete** | **public void OnSetPreferredProvidersComplete(**  **IMbnInterface** *newInterface***,**  **uint** *requestID***,**  **uint** *status*  **);** |
| **OnSubscriberInformationChange** | **public void OnSubscriberInformationChange(**  **IMbnInterface** *newInterface*  **);** |

IMbnInterfaceManager Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **GetInterface** | **public IMbnInterface GetInterface (**  **string** *interfaceID*  **);** |
| **GetInterfaces** | **public System.Array GetInterfaces ();** |

IMbnInterfaceManagerEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnInterfaceArrival** | **public void OnInterfaceArrival(**  **IMbnInterface** *newInterface*  **);** |
| **OnInterfaceRemoval** | **public void OnInterfaceRemoval(**  **IMbnInterface** *oldInterface*  **);** |

IMbnPin Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **Change** | **public void Change(**  **string** *pin***,**  **string** *newPin***,**  **out uint** *requestID*  **);** |
| **Disable** | **public void Disable(**  **string** *pin***,**  **out uint** *requestID*  **);** |
| **Enable** | **public void Enable(**  **string** *pin***,**  **out uint** *requestID*  **);** |
| **Enter** | **public void Enter(**  **string** *pin***,**  **out uint** *requestID*  **);** |
| **GetPinManager** | **public IMbnPinManager GetPinManager();** |
| **PinFormat** | [**public MBN\_PIN\_FORMAT PinFormat {** *get***; }**](#RANGE!_WWAN_PIN_TYPE,_WWAN_PIN_MODE,_WW)**;** |
| **PinLengthMax** | **public uint PinLengthMax { *get*; };** |
| **PinLengthMin** | **public uint PinLengthMin { *get*; };** |
| **PinMode** | [**public MBN\_PIN\_MODE PinMode { *get*; }**](#RANGE!_WWAN_PIN_TYPE,_WWAN_PIN_MODE,_WW)**;** |
| **PinType** | [**public MBN\_PIN\_TYPE PinType {** *get***; }**](#RANGE!_WWAN_PIN_TYPE,_WWAN_PIN_MODE,_WW)**;** |
| **Unblock** | **public void Unblock(**  **string** *puk***,**  **string** *newPin***,**  **out uint** *requestID*  **);** |

IMbnPinEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnChangeComplete** | **public void OnChangeComplete(**  **IMbnPin** *pin***,**  **ref MBN\_PIN\_INFO** *pinInfo***,**  **uint** *requestID***,**  **int** *status*  **);** |
| **OnDisableComplete** | **public void OnDisableComplete (**  **IMbnPin** *pin***,**  **ref MBN\_PIN\_INFO** *pinInfo***,**  **uint** *requestID***,**  **int** *status*  **);** |
| **OnEnableComplete** | **public void OnEnableComplete(**  **IMbnPin** *pin***,**  **ref MBN\_PIN\_INFO** *pinInfo***,**  **uint** *requestID***,**  **int** *status*  **);** |
| **OnEnterComplete** | **public void OnEnterComplete(**  **IMbnPin** *pin***,**  **ref MBN\_PIN\_INFO** *pinInfo***,**  **uint** *requestID***,**  **int** *status*  **);** |
| **OnUnblockComplete** | **public void OnUnblockComplete(**  **IMbnPin pin,**  **ref MBN\_PIN\_INFO pinInfo,**  **uint requestID,**  **int status**  **);** |

IMbnPinManager Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **GetPin** | **public IMbnPin GetPin (**  **MBN\_PIN\_TYPE** *pinType*  **);** |
| **GetPinList** | **public System.Array GetPinList();** |
| **GetPinState** | **public void GetPinState(**  **out uint** *requestID*  **);** |

IMbnPinManagerEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnGetPinStateComplete** | **public void OnGetPinStateComplete(**  **IMbnPinManager** *pinManager***,**  **MBN\_PIN\_INFO** *pinInfo***,**  **uint requestID, uint** *status*  **);** |
| **OnPinListAvailable** | **public void OnPinListAvailable(**  **IMbnPinManager** *pinManager*  **);** |

IMbnRadio Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **HardwareRadioState** | **public MBN\_RADIO HardwareRadioState {** *get***; }** |
| **SetSoftwareRadioState** | **public SetSoftwareRadioState(**  **MBN\_RADIO** *radioState***,**  **out uint** *requestID*  **);** |
| **SoftwareRadioState** | **public MBN\_RADIO SoftwareRadioState {** *get***; }** |

IMbnRadioEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnRadioStateChange** | **public void OnRadioStateChange(**  **IMbnRadio** *newInterface*  **);** |
| **OnSetSoftwareRadioStateComplete** | **public void OnSetSoftwareRadioComplete(**  **IMbnRadio** *newInterface***,**  **uint** *requestID***,**  **int** *Status*  **);** |

IMbnRegistration Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **GetAvailableDataClasses** | **public uint GetAvailableDataClasses();** |
| **GetCurrentDataClass** | **public uint GetCurrentDataClass();** |
| **GetPacketAttachNetworkError** | **public uint GetPacketAttachNetworkError();** |
| **GetProviderID** | **public string GetProviderID();** |
| **GetProviderName** | **public string GetProviderName();** |
| **GetRegisterMode** | **public MBN\_REGISTER\_MODE GetRegisterMode();** |
| **GetRegisterState** | **public MBN\_REGISTER\_STATE GetRegisterState();** |
| **GetRegistrationNetworkError** | **public uint GetRegistrationNetworkError();** |
| **GetRoamingText** | **public string GetRoamingText();** |
| **SetRegisterMode** | **public void SetRegisterMode(**  **MBN\_REGISTER\_MODE** *registerMode***,**  **string** *providerID***,**  **uint** *dataClass***,**  **out uint** *requestID*  **);** |

IMbnRegistrationEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnPacketServiceStateChange** | **public OnPacketServiceStateChange(**  **IMbnRegistration** *newInterface*  **);** |
| **OnRegisterModeAvailable** | **public OnRegisterModeAvailable(**  **IMbnRegistration** *newInterface*  **);** |
| **OnRegisterStateChange** | **public OnRegisterStateChange(**  **IMbnRegistration** *newInterface*  **);** |
| **OnSetRegisterModeComplete** | **public OnSetRegisterModeComplete(**  **IMbnRegistration** *newInterface***,**  **uint** *requestID***,**  **int** *status*  **);** |

IMbnServiceActivation Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **Activate** | **public void Activate(**  **System.Array** *vendorSpecificData***,**  **out uint** *requestID*  **);** |

IMbnServiceActivationEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnActivationComplete** | **public void OnActivationComplete(**  **IMbnServiceActivation** *newInterface***,**  **System.Array** *vendorSpecificData***,**  **uint** *requestID***,**  **int** *status***,**  **uint** *networkError*  **);** |

IMbnSignal Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **GetSignalError** | **public uint GetSignalError();** |
| **GetSignalStrength** | **public uint GetSignalStrength();** |

IMbnSignalEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnSignalStateChange** | **public void OnSignalStateChange(**  **IMbnSignal** *newInterface*  **);** |

IMbnSms Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **GetSmsConfiguration** | **public IMbnSmsConfiguration GetSmsConfiguration();** |
| **GetSmsStatus** | **public void GetSmsStatus(**  **out MBN\_SMS\_STATUS\_INFO** *smsStatusInfo*  **);** |
| **SetSmsConfiguration** | **public SetSmsConfiguration(**  **IMbnSmsConfiguration** *smsConfiguration***,**  **out uint** *requestID*  **);** |
| **SmsDelete** | **public SmsDelete(**  **ref MBN\_SMS\_FILTER** *smsFilter***,**  **out uint** *requestID*  **);** |
| **SmsRead** | **public SmsRead(**  **ref MBN\_SMS\_FILTER** *smsFilter***,**  **MBN\_SMS\_FORMAT** *smsFormat***,**  **out uint** *requestID*  **);** |
| **SmsSendCdma** | **public SmsSendCdma(**  **string** *address***,**  **MBN\_SMS\_CDMA\_ENCODING** *encoding***,**  **MBN\_SMS\_CDMA\_LANG** *language***,**  **uint** *sizeInCharacters***,**  **System.Array** *message***,**  **out uint** *requestID*  **);** |
| **SmsSendCdmaPdu** | **public SmsSendCdmaPdu(**  **System.Array** *message***,**  **out uint** *requestID*  **);** |
| **SmsSendPdu** | **public SmsSendPdu(**  **string** *pduData***,**  **byte** *size***,**  **out uint** *requestID*  **);** |

IMbnSmsConfiguration Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **CdmaShortMsgSize** | **public uint CdmaShortMsgSize {***get***; };** |
| **MaxMessageIndex** | **public uint MaxMessageIndex {** *get***; };** |
| **ServiceCenterAddress** | **public string ServiceCenterAddress{** *set***;** *get***; };** |
| **SmsFormat** | **public MBN\_SMS\_FORMAT SmsFormat{***set***;** *get***; };** |

IMbnSmsEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnSetSmsConfigurationComplete** | **public void OnSetSmsConfigurationComplete(**  **IMbnSms** *sms***,**  **uint** *requestID***,**  **int** *status*  **);** |
| **OnSmsConfigurationChange** | **public void OnSmsConfigurationChange(**  **IMbnSms** *sms*  **);** |
| **OnSmsDeleteComplete** | **public void OnSmsDeleteComplete(**  **IMbnSms** *sms***,**  **uint** *requestID***,**  **int** *status*  **);** |
| **OnSmsNewClass0Message** | **public void OnSmsNewClass0Message(**  **IMbnSms** *sms***,**  **MBN\_SMS\_FORMAT** *smsFormat***,**  **System.Array** *readMsgs*  **);** |
| **OnSmsReadComplete** | **public void OnSmsReadComplete(**  **IMbnSms** *sms***,**  **MBN\_SMS\_FORMAT** *smsFormat***,**  **System.Array** *readMsgs***,**  **bool** *moreMessage***,**  **uint** *requestID***,**  **int** *status*  **);** |
| **OnSmsSendComplete** | **public void OnSmsSendComplete(**  **IMbnSms** *sms***,**  **uint** *requestID***,**  **int** *status*  **);** |
| **OnSmsStatusChange** | **public void OnSmsStatusChange(**  **IMbnSms** *sms*  **);** |

IMbnSmsReadMsgPdu Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **Index** | **public uint Index{** *get***; };** |
| **Message** | **public BYTE[] Message {** *get***; };** |
| **PduData** | **public string PduData{***get***; };** |
| **Status** | **public MBN\_MSG\_STATUS Status{** *get***; };** |

IMbnSmsReadMsgTextCdma Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **Address** | **public string Address{** *get***; };** |
| **EncodingID** | **public MBN\_SMS\_CDMA\_ENCODING EncodingID {** *get***; };** |
| **Index** | **public uint Index{** *get***; };** |
| **LanguageID** | **public MBN\_SMS\_CDMA\_LANG LanguageID {** *get***; };** |
| **Message** | **public System.Array Message{** *get***; };** |
| **SizeInCharacters** | **public uint SizeInCharacters{** *get***; };** |
| **Status** | **public MBN\_MSG\_STATUS Status{** *get***; };** |
| **Timestamp** | **public string Timestamp {** *get***; };** |

IMbnSubscriberInformation Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **SimIccID** | **public string SimIccID(*get*;);** |
| **SubscriberID** | **public string SubscriberID(*get*;);** |
| **TelephoneNumbers** | **public System.Array TelephoneNumbers(*get*;);** |

IMbnVendorSpecificOperation Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **SetVendorSpecific** | **public void SetVendorSpecific (**  **System.Array** *vendorSpecificData***,**  **out uint** *requestID*  **);** |

IMbnVendorSpecificEvents Interface

|  |  |
| --- | --- |
| Method | C# Signature |
| **OnEventNotification** | **public void OnEventNotification(**  **IMbnVendorSpecific** *newInterface***,**  **System.Array** *vendorSpecificData*  **);** |
| **OnSetVendorSpecificComplete** | **public void OnSetVendorSpecificComplete(**  **IMbnVendorSpecific** *newInterface***,**  **System.Array** *vendorSpecificData***,**  **uint** *requestID*  **);** |