MBIM-Based Mobile Broadband Requirements for Windows

April 23, 2012

Abstract

This paper provides information about Windows specific expectations in Windows 8 Consumer Preview for mobile broadband devices that are implemented based on the Mobile Broadband Interface Model (MBIM) specification.

This information also applies to the following operating systems:   
 Windows 8 Consumer Preview

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

The current version of this paper is maintained at [MBIM-Based Mobile Broadband Requirements for Windows](http://msdn.microsoft.com/en-us/windows/hardware/hh918600).

**Disclaimer**: This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet website references, may change without notice. Some information relates to pre-released product which may be substantially modified before it’s commercially released. Microsoft makes no warranties, express or implied, with respect to the information provided here. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2012 Microsoft. All rights reserved.

License

**Microsoft Corporation Technical Documentation License Agreement (Standard)**

**READ THIS!** THIS IS A LEGAL AGREEMENT BETWEEN MICROSOFT CORPORATION ("MICROSOFT") AND THE RECIPIENT OF THESE MATERIALS, WHETHER AN INDIVIDUAL OR AN ENTITY ("YOU"). IF YOU HAVE ACCESSED THIS AGREEMENT IN THE PROCESS OF DOWNLOADING MATERIALS ("MATERIALS") FROM A MICROSOFT WEB SITE, BY CLICKING "I ACCEPT", DOWNLOADING, USING OR PROVIDING FEEDBACK ON THE MATERIALS, YOU AGREE TO THESE TERMS. IF THIS AGREEMENT IS ATTACHED TO MATERIALS, BY ACCESSING, USING OR PROVIDING FEEDBACK ON THE ATTACHED MATERIALS, YOU AGREE TO THESE TERMS.

1. For good and valuable consideration, the receipt and sufficiency of which are acknowledged, You and Microsoft agree as follows:

(a) If You are an authorized representative of the corporation or other entity designated below ("**Company**"), and such Company has executed a Microsoft Corporation Non-Disclosure Agreement that is not limited to a specific subject matter or event ("**Microsoft NDA**"), You represent that You have authority to act on behalf of Company and agree that the Confidential Information, as defined in the Microsoft NDA, is subject to the terms and conditions of the Microsoft NDA and that Company will treat the Confidential Information accordingly;

(b) If You are an individual, and have executed a Microsoft NDA, You agree that the Confidential Information, as defined in the Microsoft NDA, is subject to the terms and conditions of the Microsoft NDA and that You will treat the Confidential Information accordingly; or

(c)If a Microsoft NDA has not been executed, You (if You are an individual), or Company (if You are an authorized representative of Company), as applicable, agrees: (a) to refrain from disclosing or distributing the Confidential Information to any third party for five (5) years from the date of disclosure of the Confidential Information by Microsoft to Company/You; (b) to refrain from reproducing or summarizing the Confidential Information; and (c) to take reasonable security precautions, at least as great as the precautions it takes to protect its own confidential information, but no less than reasonable care, to keep confidential the Confidential Information. You/Company, however, may disclose Confidential Information in accordance with a judicial or other governmental order, provided You/Company either (i) gives Microsoft reasonable notice prior to such disclosure and to allow Microsoft a reasonable opportunity to seek a protective order or equivalent, or (ii) obtains written assurance from the applicable judicial or governmental entity that it will afford the Confidential Information the highest level of protection afforded under applicable law or regulation. Confidential Information shall not include any information, however designated, that: (i) is or subsequently becomes publicly available without Your/Company’s breach of any obligation owed to Microsoft; (ii) became known to You/Company prior to Microsoft’s disclosure of such information to You/Company pursuant to the terms of this Agreement; (iii) became known to You/Company from a source other than Microsoft other than by the breach of an obligation of confidentiality owed to Microsoft; or (iv) is independently developed by You/Company. For purposes of this paragraph, "Confidential Information" means nonpublic information that Microsoft designates as being confidential or which, under the circumstances surrounding disclosure ought to be treated as confidential by Recipient. "Confidential Information" includes, without limitation, information in tangible or intangible form relating to and/or including released or unreleased Microsoft software or hardware products, the marketing or promotion of any Microsoft product, Microsoft's business policies or practices, and information received from others that Microsoft is obligated to treat as confidential.

2. You may review these Materials only (a) as a reference to assist You in planning and designing Your product, service or technology ("Product") to interface with a Microsoft Product as described in these Materials; and (b) to provide feedback on these Materials to Microsoft. All other rights are retained by Microsoft; this agreement does not give You rights under any Microsoft patents. You may not (i) duplicate any part of these Materials, (ii) remove this agreement or any notices from these Materials, or (iii) give any part of these Materials, or assign or otherwise provide Your rights under this agreement, to anyone else.

3. These Materials may contain preliminary information or inaccuracies, and may not correctly represent any associated Microsoft Product as commercially released. All Materials are provided entirely "AS IS." To the extent permitted by law, MICROSOFT MAKES NO WARRANTY OF ANY KIND, DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, AND ASSUMES NO LIABILITY TO YOU FOR ANY DAMAGES OF ANY TYPE IN CONNECTION WITH THESE MATERIALS OR ANY INTELLECTUAL PROPERTY IN THEM.

4. If You are an entity and (a) merge into another entity or (b) a controlling ownership interest in You changes, Your right to use these Materials automatically terminates and You must destroy them.

5. You have no obligation to give Microsoft any suggestions, comments or other feedback ("Feedback") relating to these Materials. However, any Feedback you voluntarily provide may be used in Microsoft Products and related specifications or other documentation (collectively, "Microsoft Offerings") which in turn may be relied upon by other third parties to develop their own Products. Accordingly, if You do give Microsoft Feedback on any version of these Materials or the Microsoft Offerings to which they apply, You agree: (a) Microsoft may freely use, reproduce, license, distribute, and otherwise commercialize Your Feedback in any Microsoft Offering; (b) You also grant third parties, without charge, only those patent rights necessary to enable other Products to use or interface with any specific parts of a Microsoft Product that incorporate Your Feedback; and (c) You will not give Microsoft any Feedback (i) that You have reason to believe is subject to any patent, copyright or other intellectual property claim or right of any third party; or (ii) subject to license terms which seek to require any Microsoft Offering incorporating or derived from such Feedback, or other Microsoft intellectual property, to be licensed to or otherwise shared with any third party.

6. Microsoft has no obligation to maintain confidentiality of any Microsoft Offering, but otherwise the confidentiality of Your Feedback, including Your identity as the source of such Feedback, is governed by Your NDA.

7. This agreement is governed by the laws of the State of Washington. Any dispute involving it must be brought in the federal or state superior courts located in King County, Washington, and You waive any defenses allowing the dispute to be litigated elsewhere. If there is litigation, the losing party must pay the other party’s reasonable attorneys’ fees, costs and other expenses. If any part of this agreement is unenforceable, it will be considered modified to the extent necessary to make it enforceable, and the remainder shall continue in effect. This agreement is the entire agreement between You and Microsoft concerning these Materials; it may be changed only by a written document signed by both You and Microsoft.



Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| April 23, 2012 | First publication | | | |

Contents

[Introduction 5](#_Toc322949405)

[Terms 6](#_Toc322949406)

[Union Function Descriptors 6](#_Toc322949407)

[MBIM Backward-Compatible Functions 9](#_Toc322949408)

[Identity Morphing 10](#_Toc322949409)

[Solution Overview 10](#_Toc322949410)

[Logical configurations 10](#_Toc322949411)

[Goals of the solution 11](#_Toc322949412)

[Assumptions 11](#_Toc322949413)

[Supported transitions 11](#_Toc322949414)

[Transition details 12](#_Toc322949415)

[Solution Details 15](#_Toc322949416)

[Compatible IDs 16](#_Toc322949417)

[Appendix: Microsoft OS Descriptors 20](#_Toc322949418)

[String Descriptor 20](#_Toc322949419)

[Structure of the OS string 20](#_Toc322949420)

[Retrieving the OS string descriptor 21](#_Toc322949421)

[Verifying the integrity of the OS descriptor 21](#_Toc322949422)

[Microsoft OS string descriptor constraints 22](#_Toc322949423)

[Feature Descriptor 22](#_Toc322949424)

[Retrieving an OS feature descriptor 22](#_Toc322949425)

[Microsoft OS feature descriptor constraints 23](#_Toc322949426)

[Structure of the feature descriptor 23](#_Toc322949427)

[Example 25](#_Toc322949428)

[References 27](#_Toc322949429)

# 

# Introduction

In Windows 8, Microsoft will provide an inbox class driver, referred to as MBCD, for MBIM functions. Microsoft already provides an inbox driver, USBCCGP, for composite devices. This document states the requirements for mobile broadband devices to load USBCCGP and MBCD in Windows 8.

Mobile broadband composite devices that use WMC UFD for grouping interfaces into functions should implement Microsoft OS descriptors to load USBCCGP on Windows 8 and instruct USBCCGP to parse WMC UFD to create functions. Mobile broadband composite devices that use Interface Association Descriptors (IADs) for grouping interfaces into functions do not need to implement Microsoft OS descriptors to load USBCCGP.

MBIM functions that are backward compatible should implement Microsoft OS descriptors to load MBCD. MBIM functions that are not backward compatible do not need to implement Microsoft OS descriptors to load MBCD.

Mobile broadband devices that exhibit identity morphing should also implement Microsoft OS descriptors.

These scenarios are discussed in more detail throughout this document. The following table summarizes all of the Microsoft OS compatible IDs mentioned in this document. For more information see [Microsoft OS Descriptors](http://www.microsoft.com/whdc/connect/usb/os_desc.mspx).

Table 1. Microsoft OS compatible IDs

|  |  |  |
| --- | --- | --- |
| Microsoft OS Compatible ID | Microsoft OS Sub Compatible ID | Required for Scenario |
| “CDC\_WMC” |  | Loading USBCCGP on composite devices that use WMC UFD for grouping interfaces into functions |
| “MBIM” |  | Loading MBCD on MBIM backward-compatible function |
| “ALTRCFG” | Configuration number in ASCII | Identity morphing with IADs |
| “WMCALTR” | Configuration number in ASCII | Identity morphing with WMC UFD |

# Terms

The following terminology is used throughout this white paper.

Table 2. Terminology

|  |  |
| --- | --- |
| Term | Description |
| MBIM | Mobile Broadband Interface Model, a USB Device Working Group (DWG) specification for mobile broadband devices. |
| MBIM function | A USB function within a USB device that is compliant with the MBIM specification. |
| Mobile broadband device | A USB device that is either single function or multifunctional. In the single function case, the function should be an MBIM function. In the multifunction case, one of the functions is the MBIM function. This may also be a multiconfiguration device in which at least one of the configurations contains the MBIM function. |
| NCM2 | The earlier name for the MBIM specification. Some diagrams still refer to the MBIM functions as NCM2 functions. |
| Virtual CD-ROM | A CD-ROM function that does not have a physical CD-ROM drive. |
| IAD | Interface association descriptors (IADs) used to group interfaces into functions. |
| WMC UFD | Union function descriptors (UFDs) described in the Wireless Mobile Communication (WMC) specifications. UFDs are used to group interfaces into functions. This is an alternative to using IADs. |
| Morphing | The ability of a USB device to expose a different set of USB functions than what is currently exposed. |
| Driver | Software required by Windows to work with a USB function. |
| Inbox driver | A driver supplied by Microsoft for USB functions. These drivers are present in Windows. |
| IHV driver | A driver supplied by the independent hardware vendor (IHV) for USB functions that do not have inbox drivers. |
| IHV driver package | A collection of all IHV drivers supplied by the IHV. |
| USBHUB | A Microsoft USB hub driver. |
| USBCCGP | A Microsoft driver for USB composite devices. |
| MBCD | Mobile Broadband Class Driver, the inbox driver in Windows 8 for USB functions that conform to the MBIM specification. |

# Union Function Descriptors

Mobile broadband devices that implement UFDs have Device Class / Subclass / Protocol of 2 / 0 / 0 as required for CDC devices. This prevents Windows from loading USBCCGP on the device. For information on how Windows loads USBCCGP on composite devices, see [USB Generic Parent Driver (Usbccgp.sys)](http://msdn.microsoft.com/en-us/library/ff539234(v=VS.85).aspx).

To allow Windows to load USBCCGP, the device needs to report a Microsoft OS compatible ID of “CDC\_WMC” when the device is not configured. After detecting the compatible ID of “CDC\_WMC”, Windows loads USBCCGP, and USBCCGP sets the configuration on the device to 1. USBCCGP will then query again for the Microsoft OS compatible IDs. This time, however, the device should not report the Microsoft OS compatible ID of “CDC\_WMC”. The device may report Microsoft OS compatible IDs for functions in the selected configuration.

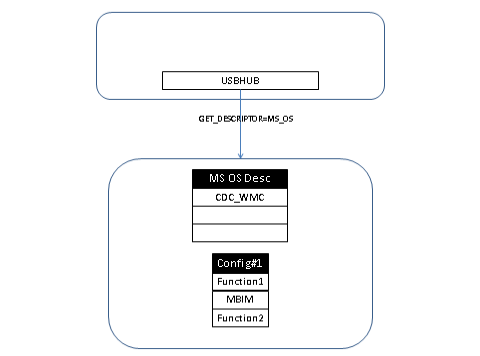


Figure . USBHUB queries for the Microsoft OS descriptor when the device is not configured

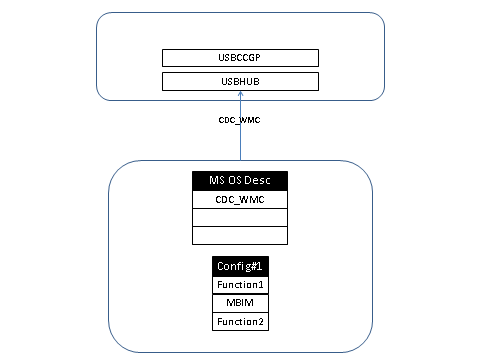


Figure . The device responds with "CDC\_WMC", which causes Windows to load USBCCGP

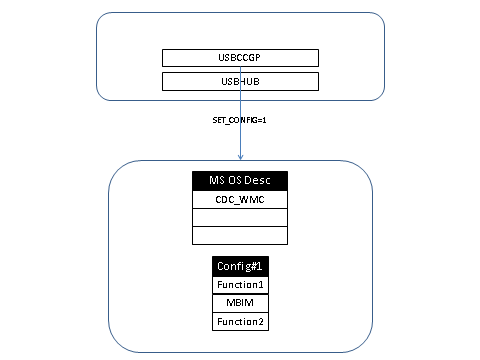


Figure . USBCCGP selects Configuration #1 on the device

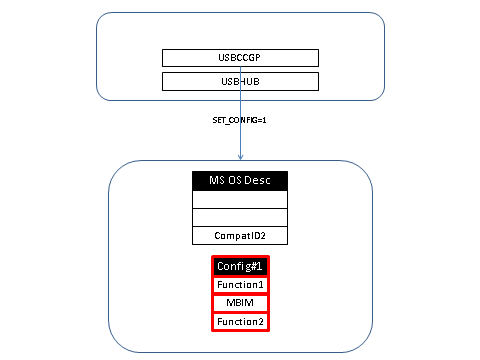


Figure . The device selects the configuration and morphs the list of compatible IDs. The device may include ComaptID2, which is necessary for Function2.

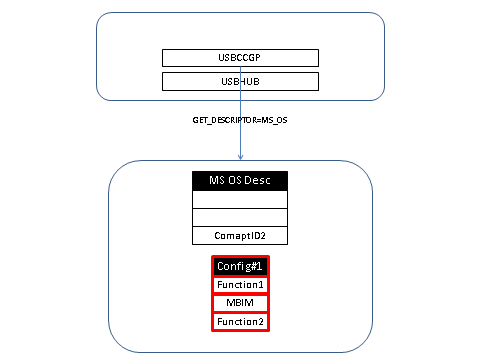


Figure . After loading, USBCCGP queries for Microsoft OS compatible IDs again

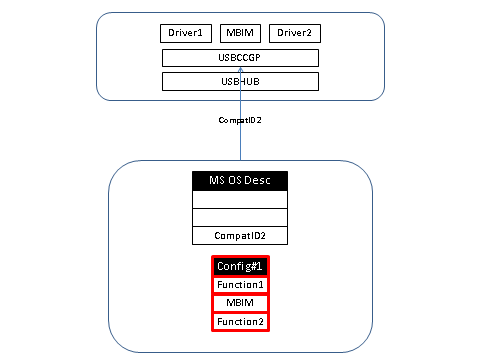


Figure . The device reports any compatible ID that it has for its function. USBCCGP then creates child device nodes for each function in the device.

# MBIM Backward-Compatible Functions

MBIM functions that are backward compatible with the NCM 1.0 specification will come up as NCM 1.0 functions by default. Mobile broadband devices that consist of an MBIM backward-compatible function should report a Microsoft OS compatible ID of “MBIM” for the MBIM function. This allows Windows 8 to detect the NCM 1.0 function as the MBIM function and load MBCD as the function driver.

# Identity Morphing

Mobile broadband USB dongle solutions eliminate the need for distributing the driver package for mobile broadband and other IHV functions through separate media (such as CD-ROM) by having a storage function in the USB device itself that contains the driver package.

Upon first time insertion of such a device in Windows, the device presents itself as mass storage, which results in Windows “Auto Play” appearing to the user. At this point, the device exposes no other functions to the host except the mass storage function to prevent the other functions appearing to the user as non-functional due to missing driver software. The user can run the IHV-supplied software that installs the driver package. In addition to installing the driver package, the IHV-supplied software also morphs the device to expose the other functions to the user.

Mobile broadband devices that use the previously described mechanism when inserted in Windows 8 would come up as mass storage. Because Windows 8 has native support for mobile broadband functions that conform to the MBIM specification, installation of the driver package is not necessary for the user to use the mobile broadband function. To allow the user to use the mobile broadband device without the need to install the driver package on Windows 8, Microsoft has proposed a solution. This document provides guidance to IHVs on how to implement this solution for Windows 8.

Mobile broadband devices that exhibit morphing behavior are referred to as *morphing devices* throughout this document.

## Solution Overview

The proposed solution maps the morphing device’s USB configuration to a set of USB functions. At any point in time, a single set of functions (by way of a configuration) are exposed to the host. The solution achieves morphing by switching between these configurations.

### Logical configurations

The functions present in the device are grouped into the following logical sets.

Table 3. Logical Set of Functions

|  |  |
| --- | --- |
| Logical Set of Functions | Description |
| Windows-7-Configuration | Configuration selected by Windows 7 and older versions of Windows when the *morphing device* is inserted into the host for the first time. |
| Windows-8-Configuration | Configuration selected by Windows 8 when the *morphing device* is inserted into the host. |
| IHV-NCM-1.0-Configuration | Configuration selected by the IHV software installed on Windows 7 and older versions of Windows after the user installs the driver package. |
| IHV-NCM-2.0-Configuration | Configuration selected by the IHV software installed on Windows 8 after the user installs the driver package. |

The following table shows the USB configurations listed in the previous table along with possible interfaces and functions. Additional requirements for each configuration are listed later.

Table 4. USB Configurations

|  |  |  |  |
| --- | --- | --- | --- |
| Configuration 1  (Windows-7-Configuration) | Configuration 2  (IHV–NCM-10-Configuration) | Configuration 3  (Windows-8- Configuration) | Configuration 4  (IHV–NCM-20- Configuration) |
| Mass CD-ROM  Mass SD | Mass CD-ROM  Mass-SD  NCM1.0  Modem  TV  GPS  FP  PC/SC smart card  Voice  Diag | Mass CD-ROM  Mass –SD  MBIM | Mass CD-ROM  Mass-SD  NCM2.0  Modem  TV  GPS  FP  PC/SC smart card  Voice  Diag |

### Goals of the solution

* In Windows 8, users should not have to perform extra steps for installing driver packages to use the mobile broadband function on *morphing devices* that conform to the MBIM specification.
* In Windows 7, users need to perform the extra step of installing driver packages before being able to use the mobile broadband function on *morphing devices*.
* In Windows 8, users need to perform the extra step of installing driver packages before being able to use IHV functions on *morphing devices* that do not have inbox drivers.

### Assumptions

MBIM also includes backward compatibility for NCM 1.0.

### Supported transitions

For Windows 8

Not-Configured 🡪 Windows-8-Configuration

Windows-8-Configuration 🡪 IHV-NCM-2.0-Configuration

For Windows 7

Not-Configured 🡪 Windows-7-Configuration

Windows-7-Configuration 🡪 IHV–NCM-1.0-Configuration

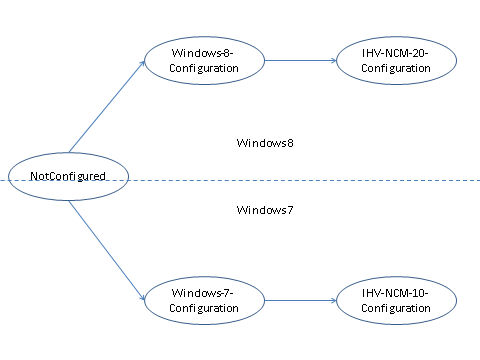


Figure 7. The configuration transition paths for Windows 7 and Windows 8

Note that any transition not shown previously is notsupported.

### Transition details

This section provides more detail about the transitions. Consider a sample USB *morphing device* with the following functions in its configurations.

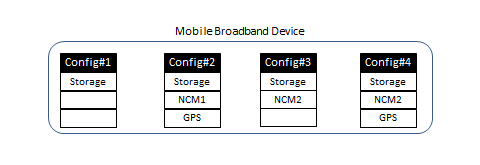


Figure 8. USB device with multiple functions

Windows 8

##### Windows-8-Configuration

When the *morphing device* is plugged into a computer running Windows 8, the Windows-8-Configuration would be selected, which exposes the MBIM function. The Windows 8 Mobile Broadband Class Driver (MBCD) will be loaded on the MBIM function. In the following example, Configuration 3 is the Windows-8-Configuration containing the MBIM function.

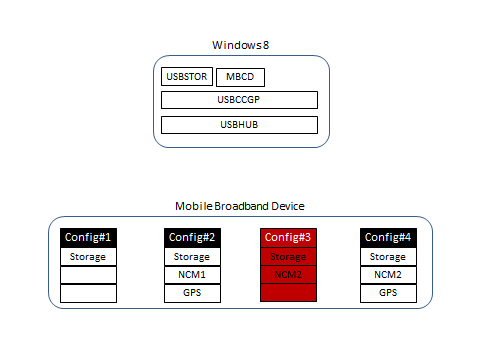


Figure 9. Driver stack and device configuration on Windows 8 after device is plugged in

##### IHV-NCM-2.0-Configuration

In the Windows-8-Configuration, the *morphing device* also has a mass storage function that will allow the user to install the IHV driver package. After installation of the driver package from the mass storage function, the device will morph to expose the functions in the IHV-NCM-2.0-Configuration. This configuration has an additional IHV function such as GPS, diagnostics, and so on. Configuration 4 in the following diagram represents the IHV-NCM-2.0-Configuration.

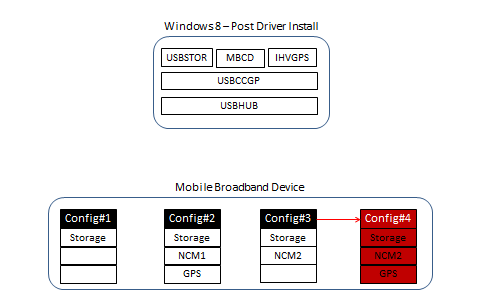


Figure 10. Driver stack and device configuration on Windows 8 after user installs IHV driver package

Windows 7

##### Windows-7-Configuration

When the *morphing device* is plugged into a computer running Windows 7 or an earlier version of Windows, the Windows-7-Configuration would be selected, which exposes the mass storage function. This will allow the user to install the IHV driver package from the mass storage function.

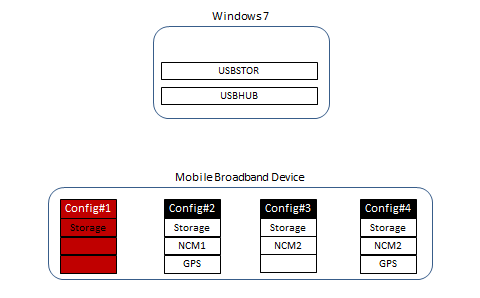
In the following example, Configuration 1 is the Windows-7-Configuration.

Figure 11. Driver stack and device configuration on Windows 7 when the user has not installed the IHV driver package

##### IHV-NCM-1.0-Configuration

In Windows 7, the user can install the driver package from the mass storage function. Along with installing the driver software, the IHV software will also morph the device from the Windows-7-Configuration to the IHV-NCM-1.0-Configuration.

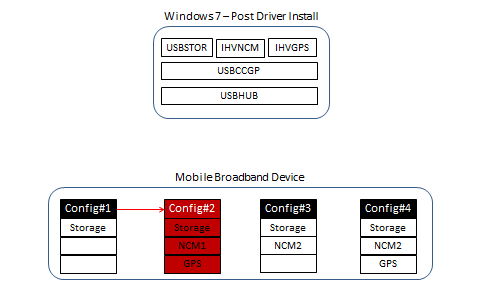


Figure 12. Driver stack and device configuration in Windows 7 after user installs IHV driver package

## Solution Details

Configuration requirements

The order of the functions across transitions in Windows 8 needs to be maintained. For example, if MBIM is the third function in the Windows-8-Configuration, it should also be the third function in the IHV-NCM-2.0-Configuration.

##### Windows-7-Configuration

The Windows-7-Configuration should be the first configuration in the *morphing device*. This configuration should have the mass storage function as one of the functions. Windows 8 will not select this configuration. In Windows 7 and earlier versions of Windows, the Windows-7-Configuration is the default configuration selected. This configuration is used to expose a USB mass storage function where IHVs put their driver package, which allows users to install the IHV’s driver.

##### Windows-8-Configuration

The Windows-7-Configuration exposes the MBIM function as one of the functions on which MBCD is loaded. In Windows 8, the value of this configuration is used in the *subCompatibleID* value returned to USBCCGP. USBCCGP selects this configuration when it is loaded. The Windows-8-Configuration should be either Configuration 2, 3, or 4. No other configuration is supported as the Windows-8-Configuration. This configuration also exposes the mass storage function as the first function to allow a user to install the IHV’s driver package.

##### IHV-NCM-2.0-Configuration

The IHV-NCM-2.0-Configuration exposes IHV-specific functions along with MBIM and mass storage functions. This configuration is not set or used by Windows. The IHV software, after installation by the user, can morph to this configuration. Note that the order of the functions in this configuration should be the same as in the Windows-8-Configuration. Although extra functions can be added to the Windows-8-Configuration, the existing functions should be retained in the same order.

##### IHV-NCM-1.0-Configuration

The IHV-NCM-1.0-Configuration exposes IHV-specific functions along with NCM 1.0 and mass storage functions. This configuration is not set or used by Windows 8. This configuration is used only in Windows 7 and earlier versions of Windows after the IHV software is installed by the user. The IHV software morphs the *morphing device* from the Windows-7-Configuration to this configuration.

### Compatible IDs

Compatible IDs are 8-character or smaller strings used by the device to indicate the driver loading preference to Windows. Devices can define compatible IDs by using Microsoft OS descriptors. Compatible and subcompatible IDs apply to individual functions. Each configuration can have a separate set of compatible IDs, which map to the set of functions within that configuration. Although compatible and subcompatible IDs apply to individual functions, the *morphing device* can have a single compatible ID when no configuration is selected. This compatible and subcompatible ID logically applies to the whole *morphing device*.

Loading USBCCGP

In Windows 8, a USBCCGP driver is required to automatically select the Windows-8-Configuration on the *morphing device*.

To load the USBCCGP driver, the *morphing device* needs to report the following compatible and subcompatible IDs when no configuration is selected on the *morphing device*.

* If the *morphing device* uses IADs for grouping interfaces into functions, the compatible ID should be reported as “ALTRCFG” and the subcompatible ID as the number of the Windows-8-Configuration.
* If the *morphing device* uses WCM UFDs for grouping interfaces into functions, the compatible ID should be reported as “WMCALTR” and the subcompatible ID as the number of the Windows-8-Configuration.

For example, if the Windows-8-Configuration is Configuration 3, the subcompatible ID would be “3” in both of these cases.

Morphing compatible IDs

During USB device enumeration, USBHUB queries the *morphing device* for the compatible ID when no configuration is selected on the *morphing device*. The *morphing device* should return the compatible and subcompatible ID used to load USBCCGP, as described in the previous section.

After USBHUB loads USBCCGP, USBCCGP selects the configuration indicated by the subcompatible ID reported earlier. USBCCGP then queries the compatible and subcompatible ID a second time. At this point, the *morphing device* should return the compatible and subcompatible IDs for the configuration that is currently selected. Therefore, after USBCCGP loads and selects a particular configuration, the *morphing device* needs to morph the compatible and subcompatible IDs that are reported. The *morphing device* must not report the compatible and subcompatible IDs that are used to load USBCCGP after a configuration is selected.

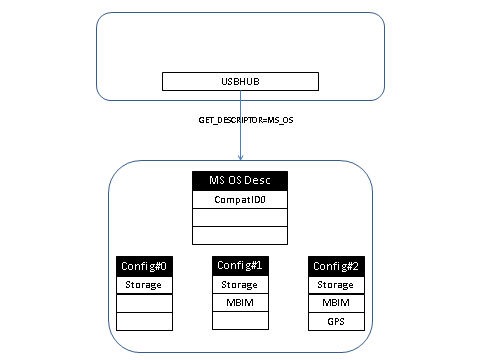


Figure 13. USBHUB querying the Microsoft OS descriptor from the device during enumeration

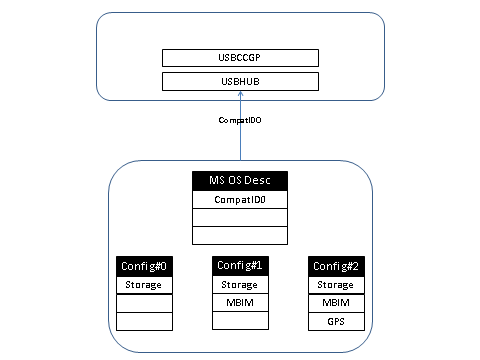


Figure 14. Device returns CompatId in the not-configured state. This CompatId is used to load USBCCGP.

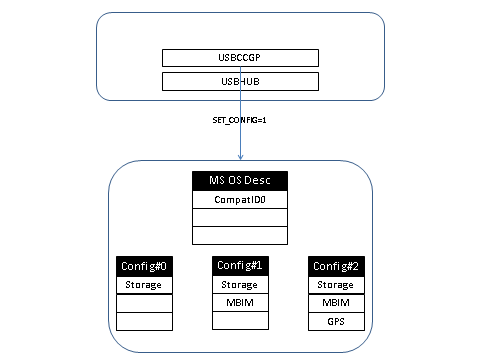


Figure 15. USBCCGP selects the configuration reported in the subcompatible ID

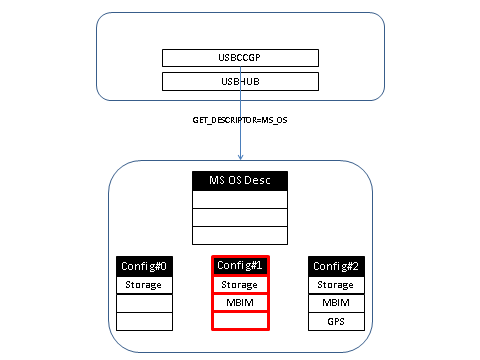


Figure 16. Device morphs its Microsoft OS descriptor based on the new configuration. USBCCGP queries for the Microsoft OS descriptor.

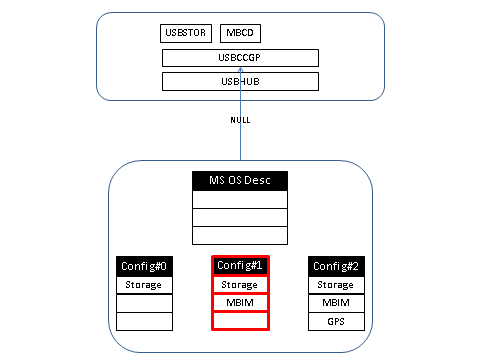


Figure 17. Device does not return any CompatID. Based on the Class / Subclass / Protocol, USBCCGP loads USBSTOR and MBCD.

# Appendix: Microsoft OS Descriptors

The Microsoft OS descriptor is broken up into the following segments:

* One Microsoft OS string descriptor
* One or more Microsoft OS feature descriptors

To support the OS descriptor, the device must implement the string descriptor.

## String Descriptor

The Microsoft OS string descriptor is a string that is stored at string index 0xEE. The format of this string is well defined.

The Microsoft OS String Descriptor is used to achieve the following objectives:

* The presence of the Microsoft OS string descriptor indicates to the operating system that the device has information embedded in it, in the form of Microsoft OS feature descriptors.
* The Microsoft OS string descriptor has an embedded signature field that is used to differentiate it from random strings that might happen to be on a device at string index 0xEE.
* The Microsoft OS string descriptor also has an embedded version number that allows for future revisions of the Microsoft OS descriptor.

Only one Microsoft OS string descriptor is stored on a device. The following sections describe the structure of the Microsoft OS string descriptor and its retrieval procedure.

### Structure of the OS string

Here is the structure of the string descriptor.

Table 5. String Descriptor Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Length (Bytes) | Value | Description |
| bLength | 1 | 0x12 | Length of the descriptor |
| bDescriptorType | 1 | 0x03 | String descriptor |
| qwSignature | 14 | ‘MSFT100’ | Signature field (4D00530046005400310030003000) |
| bMS\_VendorCode | 1 | Vendor Code | Vendor code to fetch other OS feature descriptors |
| bPad | 1 | 0x00 | Pad field |

The structure of the Microsoft OS string descriptor is fixed for version 1.00 and has an overall length of 18 bytes. The version number of the Microsoft OS string descriptor is listed in the *qwSignature* field.

The information stored in the *bMS\_VendorCode* field must be a single byte value. It will be used to retrieve Microsoft OS feature descriptors, and this byte value is used in the *bmRequestType* field described as follows:

### Retrieving the OS string descriptor

To retrieve the information stored in the string, a standard GET\_DESCRIPTOR request must be issued to the device. Here is the format of the request.

Table 6. Standard Get\_Descriptor String Request

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| bmRequestType | bRequest | wValue | wIndex | wLength | Data |
| 1000 0000B | GET\_DESCRIPTOR | 0x03EE | 0x0000 | 0x12 | Returns the string |

The *bmRequestType* field is a bitmap composed of three parts—direction of data transfer, descriptor type, and recipient. According to the USB specification, the value of *bmRequestType* is set to 1000 0000b (0x80).

The *bRequest* field should be set to a standard GET\_DESCRIPTOR request.

For a GET\_DESCRIPTOR request, the *wValue* field is split into two parts. The high byte stores the descriptor type and the low byte stores the descriptor index. To retrieve the Microsoft OS string descriptor, the high byte should be set to retrieve a string descriptor—0x03. Because the Microsoft OS string descriptor is always stored at index 0xEE, this string index should be stored in the lower byte of the *wValue* field.

The *wIndex* is used to store the language ID, but it must be set to zero for a Microsoft OS string descriptor.

The *wLength* field is used to indicate the length of the string descriptor to be retrieved. The device should respond to any valid range from 0x02–0xFF.

If a device does not have a valid descriptor at the corresponding address (0xEE), it will respond with a request error or stall. When devices do not respond with a stall, a single-ended zero reset will be issued to the device (to recover it, if it should go into an unknown state).

### Verifying the integrity of the OS descriptor

Because vendors are allowed to use any string ID to store information, the operating system must verify that the string stored in index 0xEE is indeed the Microsoft OS string descriptor. To verify this, the following tests will be conducted. Failure of either will inhibit retrieval of the Microsoft OS feature descriptors.

* If a vendor stores a string in index location 0xEE, the operating system will retrieve the string and query it to see if it is the Microsoft OS string. This can be verified by comparing the signature field in the string to the signature field entry specified previously. A mismatch would prevent further parsing of the string.
* The second test will include a verification of the length of the string based on the version number specified in the signature field. The version number specified (in the string “MSFT100”) is 1.00. This corresponds to an 18-byte string descriptor.

### Microsoft OS string descriptor constraints

The following constraints apply to Microsoft OS string descriptors and their retrieval:

* To store information in compliance with the Microsoft OS descriptor specification, the device must have one and only Microsoft OS string descriptor that is in compliance with the information outlined in [Microsoft OS Descriptors](http://www.microsoft.com/whdc/connect/usb/os_desc.mspx).
* A device vendor is free to use any value in the *bMS\_VendorCode* field in the Microsoft OS string descriptor.

## Feature Descriptor

A feature descriptor is a fixed-format descriptor that has been defined for a specific purpose.

### Retrieving an OS feature descriptor

To retrieve a Microsoft OS feature descriptor, a special GET\_MS\_DESCRIPTOR request needs to be issued to the device. Here is the format of the request.

Table 7. Standard device request format

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| bmRequestType | bRequest | wValue | wIndex | wLength | Data |
| 1100 0000b | GET\_MS\_DESCRIPTOR | X | Feature Index | Length | Returns descriptor |

The *bmRequestType* field is a bitmap composed of three parts—direction of data transfer, descriptor type, and recipient—and is in accordance with the USB specification. The Microsoft OS feature descriptor is a vendor-specific descriptor and the direction of data transfer is from the device to the host. Therefore, the value of *bmRequestType* is set to 1100 0000b (0xC0).

The *bRequest* field is used to indicate the format of the request. To retrieve a Microsoft OS feature descriptor, the *bRequest* field should be populated with a special GET\_MS\_DESCRIPTOR byte. The value of this byte is indicated by the *bMS\_VendorCode*, which is retrieved from the Microsoft string descriptor. For more information about the retrieval of the Microsoft OS string descriptor, see Retrieving the OS string descriptor

The *wValue* field is put to special use and is broken up into a high byte and a low byte. The high byte is used to store the interface number. This is essential for storing feature descriptors on a per interface basis, especially for composite devices, or devices with multiple interfaces. In most cases, interface 0 will be used. The low byte is used to store a page number. This feature prevents descriptors from having a size boundary of 64 KB (a limit set by the size of the *wLength* field). A descriptor will be fetched with the page value initially set to zero. If a full descriptor (size is 64 KB) is received, the page value will be incremented by one and the request for the descriptor will be sent again (this time with the incremented page value). This process will repeat until a descriptor with a size less than 64 KB is received. Note that the maximum number of pages is 255, which places a limit of 16 MB on the descriptor size.

The *wIndex* field stores the feature index number for the Microsoft OS feature descriptor being retrieved. Microsoft will maintain this list of Microsoft OS feature descriptors and indexes. To learn more about Microsoft OS feature descriptors, see to the supplementary document “[Microsoft OS Descriptors](http://www.microsoft.com/whdc/connect/usb/os_desc.mspx)”.

The *wLength* field specifies the length of the descriptor to be fetched. If the descriptor is longer than the number of bytes stated in the *wLength* field, only the initial bytes of the descriptor are returned. If it is shorter than the value specified in the *wLength* field, a short packet is returned.

If a particular OS descriptor is not present, the device will issue a request error or stall.

### Microsoft OS feature descriptor constraints

The following constraints apply to Microsoft OS feature descriptors and their retrieval:

* All Microsoft OS feature descriptors are defined and standardized. Vendors are not allowed to modify, append, or create Microsoft OS feature descriptors without direct consent from Microsoft.
* All Microsoft OS feature descriptors will have a size and version number embedded in them. These values should always report correct information to the operating system.
* A device can have more than one Microsoft OS feature descriptor embedded in its firmware.
* Some Microsoft OS feature descriptors are stored on a per-interface level, while others are unique to the device. Device-level Microsoft OS feature descriptors should set the high byte of the *wValue* field as zero.

### Structure of the feature descriptor

To identify itself as capable of supporting MBIM, a device must also support the extended configuration descriptor, which is one of the defined feature descriptors. The structure of this descriptor is as follows.

Header section

The header section stores information about the rest of the extended configuration descriptor. The *dwLength* field contains the length of the entire extended configuration descriptor. The header section also contains a version number, which will be initially set to 1.00 (0100H). Future revisions of this descriptor may be released at a later stage. Note that future versions of the extended configuration descriptor might also need to increase the number of entries in the header section, so please verify that this number is accurately stored in the device and read by the operating system.

Table 8. Extended configuration descriptor header section

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Offset | Field | Size | Value | Description |
| 0 | dwLength | 4 | Unsigned DWord | The length field describes the length of the extended configuration descriptor, in bytes. |
| 4 | bcdVersion | 2 | BCD | Extended configuration descriptor release number in Binary Coded Decimal (for example, version 1.00 is 0100H) |
| 6 | wIndex | 2 | Word | Fixed = 0x0004 |
| 8 | bCount | 1 | Byte | Total number of function sections that follow the header section = 0x01 |
| 9 | RESERVED | 7 |  | RESERVED |

Function section

The function section provides two important pieces of information. It groups consecutive interfaces that serve a similar purpose into function groups, and provides compatible and subcompatible IDs for each function.

Here is the format of the function section, including values that should be used by an MBIM device.

Table 9. Extended configuration descriptor function section

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Offset1 | Field | Size | Value | Description |
| 0 | bFirstInterfaceNumber | 1 | Byte | Starting interface number for this function = 0x00 |
| 1 | bInterfaceCount | 1 | Byte | Total number of Interfaces that must be included to from this function = 0x01 |
| 2 | compatibleID | 8 | Bytes | Compatible ID as defined in |
| 10 | subCompatibleID | 8 | Bytes | Subcompatible ID as defined in |
| 18 | RESERVED | 6 |  | RESERVED = 0 |

1Offset of the custom property section has been reset to zero. To calculate the offset of a field from the beginning of the extended configuration descriptor, add the length of the sections that precede it.

Table 10. Compatible and Subcompatible IDs based on the configuration exposing the MBIM function

|  |  |  |
| --- | --- | --- |
| bConfiguration | compatibleID | subCompatibleID |
| 2 | ALTRCFG (0x41 0x4C 0x54 0x52 0x43 0x46 0x47 0x00) | 2∅∅∅∅∅∅∅ (0x32 0x00 0x00 0x00 0x00 0x00 0x00 0x00) |
| 3 | ALTRCFG (0x41 0x4C 0x54 0x52 0x43 0x46 0x47 0x00) | 3∅∅∅∅∅∅∅ (0x33 0x00 0x00 0x00 0x00 0x00 0x00 0x00) |
| 4 | ALTRCFG (0x41 0x4C 0x54 0x52 0x43 0x46 0x47 0x00) | 4∅∅∅∅∅∅∅ (0x34 0x00 0x00 0x00 0x00 0x00 0x00 0x00) |

* bConfiguration refers to the bConfiguration value within the USB configuration descriptor of the configuration that exposes the MBIM function. bConfiguration cannot be 1 because that is the default configuration exposing only the CDROM function. bConfiguration cannot be greater than 4; that is, the MBIM function should be exposed within the first four configurations.
* compatibleID remains the same for all configurations. The subcompatible ID changes based on the configuration.

## 

## Example

This table shows a sample multiconfiguration scenario. The table lists the functions available in each configuration and the actions that different versions of the operating system takes for each of these configurations.

Table 11. Example of a multi-configuration mobile broadband device

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bConfiguration | 1 (Windows-7-Configuration) | 2 (IHV-NCM-1.0-Configuration) | 3 (Windows-8-Configuration) | 3 (IHV-NCM-2.0-Configuration) |
| **Functions exposed** | * CD-ROM * SD | * CD-ROM * SD * NCM10 * Modem * TV * GPS * FP * PC/SC smart card * Voice * Diagnostics | * CDROM * SD * MBIM | * CD-ROM * SD * NCM20 * Modem * TV * GPS * FP * PC/SC smart card * Voice * Diagnostics |

The following tables show the values used by the Microsoft OS string descriptor and the Microsoft OS extended configuration feature descriptor for the previous sample’s multiconfiguration scenario.

Table 12. Example of a multi-configuration mobile broadband device

|  |  |  |
| --- | --- | --- |
| Field | Length (Bytes) | Value |
| bLength | 1 | 0x12 |
| bDescriptorType | 1 | 0x03 |
| qwSignature | 14 | ‘MSFT100’ 0x4D 0x00 0x53 0x00 0x46 0x00 0x54 0x00 0x31 0x00 0x30 0x00 0x30 0x00 |
| bMS\_VendorCode | 1 | 0xA5 |
| bPad | 1 | 0x00 |

Table 13. Example Microsoft OS extended configuration feature descriptor header

|  |  |  |  |
| --- | --- | --- | --- |
| Offset | Field | Size | Value |
| 0 | dwLength | 4 | 16 |
| 4 | bcdVersion | 2 | 0100H |
| 6 | wIndex | 2 | 0x0004 |
| 8 | bCount | 1 | 1 |
| 9 | RESERVED | 7 |  |

Table 14. Example Microsoft OS extended configuration feature descriptor function

|  |  |  |  |
| --- | --- | --- | --- |
| Offset2 | Field | Size | Value |
| 0 | bFirstInterfaceNumber | 1 | 0 |
| 1 | bInterfaceCount | 1 | 1 |
| 2 | compatibleID | 8 | ALTRCFG 0x41 0x4C 0x54 0x52 0x43 0x46 0x47 0x00 |
| 10 | subCompatibleID | 8 | 3 |
| 18 | RESERVED | 6 |  |

2Offset of the custom property section has been reset to zero. To calculate the offset of a field from the beginning of the extended configuration descriptor, add the length of the sections that precede it.

# References

[Microsoft OS Descriptors](http://www.microsoft.com/whdc/connect/usb/os_desc.mspx)

[IADs](http://www.usb.org/developers/docs/InterfaceAssociationDescriptor_ecn.pdf)

[Enumeration of Interfaces on USB Composite Devices](http://msdn.microsoft.com/en-us/library/ff537102(v=VS.85).aspx)

[USB Generic Parent Driver (Usbccgp.sys)](http://msdn.microsoft.com/en-us/library/ff539234(v=VS.85).aspx)