

USB CDC/ECM Class Driver

For Windows 2000, XP, Vista and 7

Reference Manual

Version 1.23

September 15, 2011

Thesycon® Systemsoftware & Consulting GmbH
Werner-von-Siemens-Str. 2 · D-98693 Ilmenau · GERMANY

Tel: +49 3677 / 8462-0

Fax: +49 3677 / 8462-18

e-mail: info @ thesycon.de

<http://www.thesycon.de>

Copyright (c) 2009-2011 by Thesycon Systemsoftware & Consulting GmbH
All Rights Reserved

Disclaimer

Information in this document is subject to change without notice. No part of this manual may be reproduced, stored in a retrieval system, or transmitted in any form or by any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use, without prior written permission from Thesycon Systemsoftware & Consulting GmbH. The software described in this document is furnished under the software license agreement distributed with the product. The software may be used or copied only in accordance with the terms of the license.

Trademarks

The following trade names are referenced throughout this manual:

Microsoft, Windows, Win32, Windows NT, Windows XP, Windows Vista, Windows 7 and Visual C++ are either trademarks or registered trademarks of Microsoft Corporation.

Other brand and product names are trademarks or registered trademarks of their respective holders.

Contents

Table of Contents	6
1 Introduction	7
2 Overview	9
2.1 Platforms	9
2.2 Features	10
2.3 USB 2.0 and 3.0 Support	11
3 Architecture	13
3.1 Special Properties of the Driver	14
3.1.1 NDIS Model	14
3.1.2 Surprise Removal Dialog	14
3.1.3 Power Management	14
3.1.4 MAC Address	14
3.1.5 Statistic Counters	15
3.1.6 Multiple interfaces on one device	15
4 Driver Customization	17
4.1 Customization Overview	17
4.2 Customization Steps	18
4.3 Customizing cdcecm.inf	18
4.3.1 Configuration of Names	19
4.3.2 Configuration of Hardware ID	20
4.3.3 Update of the Driver Version	21
4.3.4 Customizing Default Driver Settings	21
4.4 Customizing Version Resources	22
4.5 Digital Signature for Windows Vista and later	23
4.5.1 Get an Authenticode Digital ID	24
4.5.2 Get the Tools for Code Signing	24
4.5.3 Create a Signature	25
4.5.4 Modified System Behavior	26
5 Driver Installation and Uninstallation	27
5.1 Driver Installation for Developers	27

5.1.1	Installing CDC ECM Driver Manually	27
5.1.2	Uninstalling CDC ECM manually	28
5.2	Installing CDC ECM driver for End Users	29
5.2.1	Installing CDC ECM driver with the PnP Driver Installer	29
5.2.2	Installing the Driver with DIFx	30
6	Debug Support	31
6.1	Event Log Entries	31
6.1.1	Clear Feature Endpoint Halt	31
6.1.2	Demo is Expired	31
6.2	Enable Debug Traces	31
7	Related Documents	35

1 Introduction

The CDCECM driver is a generic device driver for Windows. It creates a Network Interface. On the USB layer it expects a device with a Communication Device Class (CDC) Ethernet Control Model (ECM) compliant interface. The device driver supports USB 2.0 with the operating speeds full and high speed.

This document describes the architecture and the features of the CDCECM device driver. Furthermore, it includes instructions for installing and using the device driver.

The reader of this document is assumed to be familiar with the specification of the Universal Serial Bus Version 1.1 and 2.0 and the CDC ECM specification.

2 Overview

2.1 Platforms

The CDCECM driver supports the following operating system platforms:

- Windows 7
- Windows Vista
- Windows XP
- Windows 2000
- Windows Embedded Standard 7 (WES7)
- Windows Embedded Enterprise
- Windows Embedded POSReady
- Windows Embedded Server
- Windows XP embedded
- Windows Server 2008 R2
- Windows Server 2008
- Windows Server 2003
- Windows Home Server

The driver package contains 32 bit and 64 bit versions depend on operating system.

2.2 Features

The CDCECM driver provides the following features:

- **USB Support.** The ECM class driver supports USB 2.0 full and high speed and USB 1.1.
- **NDIS Support.** Exposes a network adapter interface according to the NDIS 5.1 model (NDIS 5.0 on Windows 2000).
- **Plug&Play.** The driver fully supports hot plug and play.
- **Power Management.** The driver supports the Windows power management model. The network adapter is not stopped when the PC enters standby, sleep or hibernate. A device with Auto-IP address negotiation is not required to restart the negotiation after resume from a power down state.
- **MAC Address.** By default, the driver uses the MAC address reported by the device. If the device cannot provide a MAC address, the driver can use a MAC address defined in the INF file.
- **IP Address.** The IP address of the adapter can be assigned via DHCP, Auto-IP, or manually.
- **Multiple USB Interfaces.** The ECM class driver can be used with devices that implement multiple USB interfaces. A separate network adapter instance will be created for each ECM interface. Thesycon offers a multi-interface driver that is required to build an individual device node for each interface. For more information, check out <http://www.thesycon.de> USB Multi Interface Driver.
- **Multiple USB Devices.** Multiple USB devices can be controlled by the driver at the same time.
- **WHQL Certification.** The driver conforms to Microsoft's Windows Driver Model (WDM) and NDIS, and it can be certified by Windows Hardware Quality Labs (WHQL) for all current 32-bit and 64-bit operating systems.

2.3 USB 2.0 and 3.0 Support

The CDCECM device driver supports USB 2.0 and the Enhanced Host Controller on Windows 2000, Windows XP, Windows Vista and W7. However, CDCECM has to be used on top of the driver stack that is provided by Microsoft. Thesycon does not guarantee that the CDCECM driver works in conjunction with a USB driver stack that is provided by a third party. For instance, third-party drivers are available for USB 2.0 host controllers from NEC, INTEL or VIA. Because the Enhanced Host Controller hardware interface is standardized (EHCI specification) the USB 2.0 drivers provided by Microsoft can be used with host controllers from any vendor. However, the user has to ensure that these drivers are installed.

Currently the first Extended Host Controllers are on the market. Microsoft did not release a bus driver with Windows 7 for this host controller type. The Extended Host Controller handles also full and high speed data traffic. If a customer connects your ECM device to a blue USB connector it runs with an Extended Host Controller and the related vendor provided bus driver. In the market there are different host controller drivers. Thesycon is not able to test the ECM driver with all possible host drivers that are available on the market. For that reason we cannot give any warranty that the driver works with such host controllers.

3 Architecture

Figure 1 shows the USB driver stack that is part of the Windows operating system. All drivers are embedded within the WDM layered architecture.

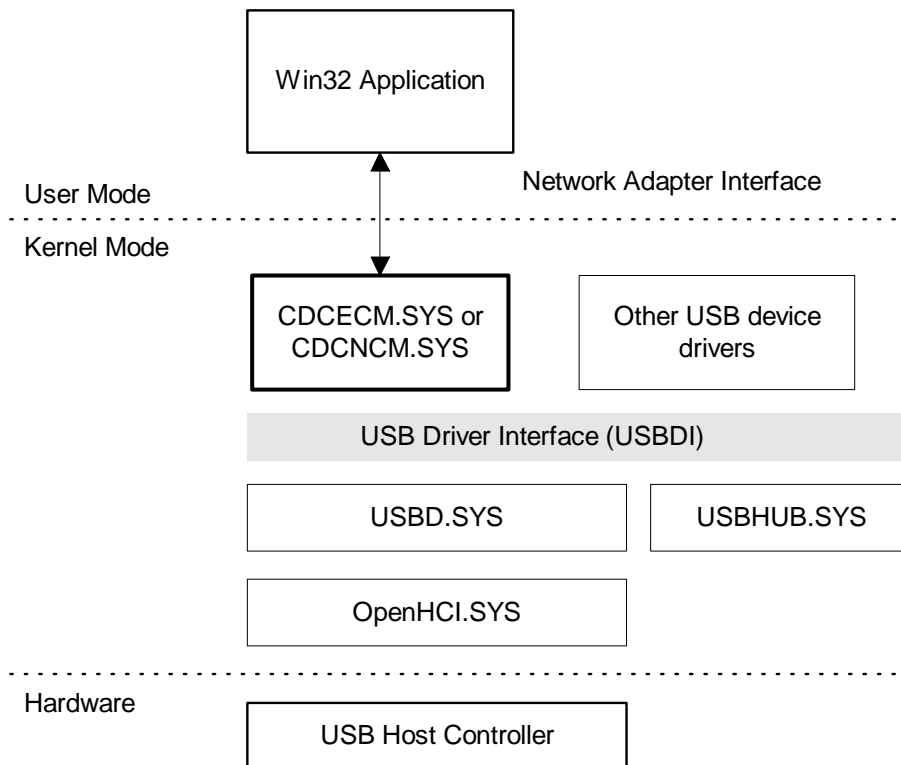


Figure 1: USB Driver Stack

The following modules are shown in Figure 1:

- USB Host Controller is the hardware component that controls the Universal Serial Bus. It also contains the USB Root Hub. There are two implementations of the host controller that support USB full speed: Open Host Controller (OHC) and Universal Host Controller (UHC). There is one implementation of the host controller that supports USB high speed: Enhanced Host Controller (EHC).
- OpenHCI.SYS is the host controller driver for controllers that conform with the Open Host Controller Interface specification. Optionally, it can be replaced by a driver for a controller that is compliant with UHCI (Universal Host Controller Interface) or EHCI (Enhanced Host Controller Interface). Which driver is used depends on the main board chip set of the computer. For instance, Intel chip sets contain Enhanced Host Controllers and Universal Host Controllers.
- USB.D.SYS is the USB Bus Driver that controls and manages all devices connected to the USB. It is provided by Microsoft as part of the operating system.
- USBHUB.SYS is the USB Hub Driver. It is responsible for managing and controlling USB Hubs.

- The CDCECM.SYS is a kernel mode driver that supports the CDC ECM protocol and provides a Network Adapter Interface.

The software interface that is provided by the operating system for use by USB device drivers is called USB Driver Interface (USBDI). It is exported by the USBD at the top of the driver stack. USBDI is an IRP-based interface. This means that each individual request is packaged into an I/O request packet (IRP), a data structure that is defined by WDM. The I/O request packets are passed to the next driver in the stack for processing and will return to the caller after completion.

3.1 Special Properties of the Driver

3.1.1 NDIS Model

Microsoft supports different versions of the NDIS model. On Windows 2000 the NDIS version 5.0 is supported. Drivers that are designed for NDIS version 5.1 cannot run on Windows 2000. On the other hand NDIS version 5.1 provides some new features on Windows XP and later. For that reason we provides two build versions of the driver. The driver for Windows 2000 has the extension 'w2k' and is compiled with the NDIS 5.0 model. The driver for Windows XP and better is compiled with the NDIS 5.1 model.

3.1.2 Surprise Removal Dialog

Windows can show plug and play devices on the PnP card in the systray and can complain if such devices are removed from the system without stopping it. The driver suppresses the entry in the systray. The device can be removed from the PC without any preparation.

3.1.3 Power Management

The driver supports the requests for power management. For that reason the adapter is not stopped if the PC enters standby, sleep or hibernate. This has the advantage that a device with AutoIP address negotiation must not restart the negotiation after the power down state of the PC. It is not recommended to use AutoIP because it takes a long time before Windows assigns a valid IP address.

3.1.4 MAC Address

The driver determines the valid network address (MAC Address) in the following way:

1. The driver reads the network address from the device. If this fails
2. The driver reads the network address from the registry. If this fails
3. The driver has a build in MAC address with the value AE:DE:48:02:01:00. This address has set the local bit and is not necessary unique.

3.1.5 Statistic Counters

The driver does not request the statistic counters from the device. It counts relevant events by it's self.

3.1.6 Multiple interfaces on one device

The CDCECM driver can be used with devices that implement multiple USB interfaces. In this case a multi-interface driver is required. This driver splits the interfaces to separate device nodes. On each device node a new driver stack can be installed.

The system provided multi-interface driver before Windows XP SP3 cannot be used together with the CDC ECM driver. The reason is that this CDC ECM interface consists of two USB interfaces. This two interfaces must be mapped to one device node. The system supplied multi-interface driver is not able to detect USB interfaces that belong together. Later versions of the system provided MI driver can handle the IAD. You should add an IAD to your configuration descriptor.

Thesycon provides a special multi-interface driver that can handle this problem in a correct way. This driver is required if you have to support systems before Windows XP SP3. This driver can be licensed separately.

4 Driver Customization

4.1 Customization Overview

The CDCECM device driver supports various features that enable you to create a customized device driver package to be shipped with an end product. Customization includes:

- Modification of the file name of the driver executable,
- Modification of text strings shown at the Windows user interface,
- Adaptation of driver behavior for a specific device.

Note that the driver package which is shipped to end users should always be customized. This is required in order to avoid potential conflicts with other products of other vendors that are also using the CDCECM device driver. Please consider the following example scenario: An end user buys product A which includes the CDCECM device driver version 2.00. The user installs this driver on his machine. At a later point in time the user buys another product of another vendor which is called B. Product B includes the CDCECM device driver version 2.30. When the user installs the device driver for Product B on his machine then a conflict will occur because another version of the driver is already installed on that machine.

There are several problems that may result from this conflict situation:

- **Driver version conflict**
We assume that during driver installation any existing device driver will be removed if the existing driver has an older version than the driver to be installed. If the existing driver is newer then no driver will be installed. In the example scenario described above this means: When product B is installed the existing driver (V2.00) will be replaced by a newer one (V2.30). The result is that both product A and product B now use the new driver. This should be fine for product B. However, product A will now run with driver version 2.30 which is critical because probably the product was never tested with that version. Thus, installation of a new product can break an existing installation of another product.
- **Driver software interface ambiguity**
If two or more different products (of different vendors) use the same driver then a conflict can arise when Windows applications open the device driver to communicate with the device. In the example scenario described above an application designed for product A could inadvertently open device B and try to configure the hardware which will probably not work.
- **Device naming ambiguity**
If two or more different products (of different vendors) use the same driver then a device name conflict could occur. Particularly, this applies to device names displayed in Device Manager. In the example scenario described above an end user could get confused if the item shown in Device Manager for product A is named identically to the item shown for product B.

The customization features of the CDCECM driver enable you to avoid all of these conflict situations. A customized driver can be considered as a specific driver for a specific product. There

will be no overlaps with (customized) CDCECM drivers shipped with other products of other vendors. In the example scenario described above, if both product A and product B are shipped with a customized driver then there will be two separate driver installations on the end user's machine. There will be two sets of driver files on hard disk and two separate drivers loaded into memory.

Note that it is possible to create a customized driver package which supports several products with similar properties, e.g. a product family of a vendor. In this case, if several products of the family are used on one machine, there will be only one set of driver files on hard disk and only one driver executable loaded into memory. A vendor can decide which of its products will be supported by a particular driver package and how many different driver packages need to be created.

To summarize the customization strategy the following rules are given:

1. A driver package provided to end users should always contain a customized driver. Do not ship the original driver provided by Thesycon together with your products.
2. If you offer a family of products, you may create a customized driver package that supports all products of this family. Windows applications shipped with the driver should be designed in such a way that all products of the family are supported. If an updated driver is delivered to end users, either as a software-only package or as part of a new product of the family, then you have to ensure that the new driver version works with all released products of the family.

In the following sections, the customization procedure is described in detail.

4.2 Customization Steps

Below, the steps needed to create a customized driver package are summarized. Some of these steps are required and some are optional.

- **Required:** Choose a new name for the driver and the .inf files. The driver package consists of the driver files `cdcecm.sys` and `cdcecm_x64.sys` as well as the INF files `cdcecm.inf` and `cdcecm_x64.inf`.

Make a private copy of the sys and inf files and rename the files to your new names. Edit the renamed .INF files to contain the new driver name. See section 4.3.1 on page 19 for detailed information.

- **Required:** Edit your .INF files to contain the correct hardware ID for your device. Refer to section 4.3.2 on page 20 for more information.
- **Optional:** Adapt default driver settings. Refer to section 4.3.4 on page 21 for more information.
- **Optional:** Edit the version resources contained in `cdcecm.sys`. See section 4.4 on page 22 for more information.

4.3 Customizing `cdcecm.inf`

The `cdcecm.inf` file is used to install the kernel-mode device driver `cdcecm.sys` on a 32 bit system. For x64 systems the files `cdcecm_x64.inf` and `cdcecm_x64.sys` are used. The .INF files

have to conform to INF file conventions defined for WDM drivers. Refer to the Windows DDK documentation [5] for more information about INF files.

The cdcecm(_x64).inf file itself can be renamed to any name of your choice. However, the file name extension has to be INF.

4.3.1 Configuration of Names

In cdcecm(_x64).inf there is a Strings section that permits to define the driver name and some text strings that will be shown at the Windows user interface.

```
[Strings]
S_Provider="Thesycon"
S_Mfg="Thesycon"
S_DiskName="CDCECM driver disk"
S_DeviceDesc="Thesycon CDC ECM network adapter"
S_DriverName="cdcecm"
S_ServiceName="cdcecm"
```

S_Provider

This variable defines the provider of the driver. You should use your company's name here.

S_Mfg

This variable defines the manufacturer of the driver. You should use your company's name here.

S_DeviceDesc

This variable defines the device description which is shown during installation. When driver installation is finished the device description is shown in Device Manager next to the item representing your device. You should use your product's name here. The name can contain spaces but the size is limited to 32 characters.

S_DiskName

This variable defines the name of your installation disk or CDROM. It should correspond to the label that is printed on the disk or CDROM. The disk name is displayed by the operating system when it requests the user to insert the installation media.

S_DriverName

This variable defines the name of the cdcecm driver executable which is cdcecm(_x64).sys by default. Note that the name is given without the .sys extension. You have to set this value to the

file name you want to use for `cdcecm.sys`. It is strongly recommended to use a vendor-specific driver name in order to avoid file naming conflicts with other drivers.

Important: The driver name must not contain spaces. If you modify `S_DriverName` then you have to modify the following sections as well:

```
[_CopyFiles_sys], [SourceDisksFiles].
```

Due to technical limitations, these sections cannot use the `S_DriverName` variable to specify the driver executable. Therefore, you have to edit them as well. Do a search for `cdcecm` to locate the lines that have to be modified.

S_ServiceName

The service name must be unique on one PC as well as the driver file name. It should contain a name equal or similar to the driver file name without any extension. Use a name without spaces and special characters.

4.3.2 Configuration of Hardware ID

Your USB device is initially enumerated by the system-provided USB bus driver. The bus driver uses vendor and product ID's from the device descriptor to create a unique hardware ID string according to the following scheme:

```
USB\VID_VVVV&PID_PPPP
```

The fields `VVVV` and `PPPP` will be replaced by the hexadecimal form of the vendor ID and the product ID. You can check the resulting hardware IDs by looking at the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USB
```

Additionally the hardware ID is displayed in the device manager under Windows XP SP2 and higher under Properties -> Details -> Hardware ID.

The `cdcecm(_x64).inf` file needs to specify the resulting hardware ID for the device within the following section:

```
[_Devices]
%S_DeviceDesc%=_Install, USB\VID_VVVV&PID_PPPP
```

Replace `VVVV` and `PPPP` by your specific values, for example:

```
[_Devices]
%S_DeviceDesc%=_Install, USB\VID_152A&PID_0001
```

Windows can distinguish between different device release numbers. The device release number is part of the device descriptor in the field `bcdDevice`. If the INF file should work with one special device release number of a device the hardware ID can be specified in the following way:

```
[_Devices]
%S_DeviceDesc%=_Install, USB\VID_152A&PID_0001&REV_ZZZZ
```

The parameter ZZZZ is the hexadecimal value of the `bcdDevice` value.

If your device contains more than one USB interface and the CDCECM driver should be loaded on an interface of your device the following hardware ID must be used:

```
[_Devices]
%S_DeviceDesc%=_Install, USB\VID_152A&PID_0001&MI_00
```

The two digits after MI are the USB interface number. If you want to load the driver for more than one interface you have to add a hardware ID for each interface, e.g.:

```
[_Devices]
%S_DeviceDesc%=_Install, USB\VID_152A&PID_0001&MI_00
%S_DeviceDesc%=_Install, USB\VID_152A&PID_0001&MI_02
```

4.3.3 Update of the Driver Version

The key word `DriverVer` is used in the Version section or in the DD Install section of the INF file. It contains the version of the driver and the release date. The release date should be modified to the release date of the customized version. Please take care of the right date format.

4.3.4 Customizing Default Driver Settings

The .INF file specifies some settings that define the default behavior of the driver. These settings are defined in the following section.

```
[_AddReg_HW]

HKR,,RxBuffers,%REG_DWORD%, 40
HKR,,TxBuffers,%REG_DWORD%, 40
HKR,,DefaultMediaState,%REG_DWORD%, 1
HKR,,OneByteTermination,%REG_DWORD%, 0
HKR,,ConfigurationIndex,%REG_DWORD%, 0
HKR,,SendClearEpHaltOnStart,%REG_DWORD%, 0
HKR,,StatisticsUpdateInterval,%REG_DWORD%, 5000
HKR,,NetworkAddress,%REG_SZ%, "AEDE48020100"
HKR,,VendorDescription,%REG_SZ%, %S_VendorDescription%
```

RxBuffers

This parameter defines number of read buffers. The default value is 40.

TxBuffers

This parameter defines number of write buffers. The default value is 40.

DefaultMediaState

If this parameter is 1 the driver reports the media state as connected. The PC can send immediately requests to the device. If this parameter is set to 0 the driver waits on a notification from the device that the media state is changed to connected. Default is 1.

OneByteTermination

If this parameter is 1 the PC driver terminates a packet that can be divided by the FIFO size without rest with an additional byte. If this parameter is zero the driver terminates such a packet with a zero length packet. Set this parameter to 1 if your hardware cannot handle zero length packets. The default value is 0. Note: Linux use the same algorithm.

ConfigurationIndex

Is the zero based USB configuration index that is activated by the driver. Some Linux gadget implementations exposes the CDC/ECM interface on index 1.

SendClearEpHaltOnStart

If this value is different from 0 the driver sends the standard command clear feature endpoint stall on each data pipe before the data transfer is started.

StatisticsUpdateInterval

This value is used to update the Ethernet statistics when the device supports it. To save bandwidth the driver requests only the parameters that are request by the PC.

NetworkAddress

With this parameter a Network address can be assigned when the device is not able to report one. When the device reports a Network address the address of the device is used. **Note:** You have to use a valid MAC address. Do not use the example above.

VendorDescription

This is the vendor description reported to the NDIS layer.

4.4 Customizing Version Resources

The `cdcecm(_x64).sys` executable includes version resources. These information will be shown in the Device Manager on the Driver Details dialog page or on the file's property page. You may want to modify the version resources to include your product's name or to modify copyright information. This can be done in a two-step procedure as described below.

1. Make a private copy of `cdcecm.sys`. It is important to use the unsigned version of the driver. If a signed version is edited the `.sys` file cannot be used any more. Use Visual Studio to open the copy of `cdcecm.sys` in resource mode. You have to select Open as Resources in the File Open dialog. Edit the version resources according to your preferences and save the modified file.

2. Open a Command Prompt window and run the UpdateChecksum.exe tool on the modified file. You have to enter the following command line:

```
UpdateChecksum cdcecm.sys
```

The program UpdateChecksum.exe is part of the CDCECM driver package.

4.5 Digital Signature for Windows Vista and later

Windows Vista and later systems have a new feature to verify a vendor of a software component. The vendor can add a signature to a software component to identify itself. This signature grants that the software was signed by the vendor and that the software was not modified after it was signed. If a signed plug and play driver is installed the first time Windows Vista and later systems show the vendor name and the user can choose if he wants to

- Abort the installation,
- Allow the installation this time or
- Always trust the vendor and allow installations from this vendor.

If the user selects "always trust this vendor" the certificate of the vendor is stored in the certification manager under Trusted Vendors. If later a plug and play driver of this vendor is installed again, the installation can be performed silently without user interactions if the driver was pre-installed before the device is connected.

On Windows Vista x64 Edition and later X64 systems it is required that the driver has a digital signature. Otherwise, the driver is not loaded on a normal system. To test a driver without signature on Windows Vista x64 and later the system can run with a kernel debugger. In this case the system loads a driver without signature. On Windows Vista 32 bit and W7 32 bit a driver without signature can be installed.

The signature of a driver is not the same as a WHQL certification. A driver can have a digital vendor signature without passing any WHQL tests.

A signature for a plug and play driver is always part of a .cat file. It is possible to add a signature to the .sys file. Please note that the signature that is attached to a .sys file modifies the file. The CRC of such a file is modified, too, and a previously generated .cat file contains an invalid CRC.

Why can Thesycon not deliver a signed driver package? The signature becomes invalid if the driver or the .INF file are modified. To install the driver on any device a customized INF file must be generated. At least the USB vendor and product ID's must match the ID's of the device. The modification of these ID's would make a signature from Thesycon invalid.

Why Thesycon cannot deliver all the tools required to create the signature? This is not possible because Microsoft does not allow re-distribution of the signing tools.

Why the Vendor that uses the CDC ECM driver should sign the driver and not Thesycon? The aim of the customization is to let the driver look like a driver that is developed by the vendor of the device. To direct support issues to the vendor and not to Thesycon it is required that the vendor of the device performs the signing.

The following sections will guide you through the process of obtaining an "Authenticode Digital ID", to maintain it, to get the tools required for signing and to create a signature for a plug and

play driver. It is strongly recommended to follow these steps. It is not possible to perform the code signing on a Windows 2000 system. Windows XP or better is required.

To get more information about code signing, read the document "Kernel-Mode Code Signing Walk through" available on the Microsoft web site.

4.5.1 Get an Authenticode Digital ID

To make sure that the owner of a digital signature key is the entity that is described with this key a so called certification authority (CA) guarantees that this information is valid. Microsoft accepts a number of CA's for the Authenticode technology. The company VerySign is one of them.

At first you have to buy a signature key pair from one of the CA's. The key is valid for a given time interval. You can select the time interval during the order of the key pair. The time interval means that the key can be used in this interval to create new signatures. A signature that was created with a time stamp is valid after the key has expired.

The signature key delivered by VerySign typically consists of three parts:

- a certificate and a public key stored in a .spc file,
- a private key stored in a .pvk file and
- a password that protects the private key.

You may read the Microsoft document "Code-Signing Best Practices" to get more information how to maintain the key.

The key must be stored in the certificate manager to use it later. To do this, it must be first translated to a personal information exchange (.pfx) file format. This can be performed by the program PVK2PFX that is part of the SDK and the WDK. This program is a command line program. Please use the following command line to create the .pfx file:

```
pvk2pfx -pvk <pvk file name>.pvk -spc <spc file name>.spc  
-pi <password> -pfx <pfx file name>.pfx
```

To import the .pfx file to the certificate manager, perform a double click to the file. This starts the Certificate Import Wizard. Enter the password and make sure the certificate is stored in the Personal certificate store.

4.5.2 Get the Tools for Code Signing

To create a .cat file the program inf2cat.exe is required. This file is part of the "Winqual submission tools". Please download this package from the Microsoft web site and extract it. The inf2cat.exe program requires the DLL's that starts with "Microsoft.Whos..." and that are part of the package.

The signtool.exe is required to attach the signature to the .cat file. This program exists in different versions with the same file name. Versions that are delivered with the .net framework and the SDK cannot be used to sign kernel mode drivers. Only the version that is part of the WDK is sufficient to create a signature for a kernel mode driver. So you have to install the WDK. You will find the signtool program under bin\SelfSign.

Kernel mode driver signing requires a cross certificate. This cross certificate must be downloaded at the Microsoft Web site "Microsoft Cross-certificates for Windows Vista Kernel Mode Code Signing". Please select the correct cross certificate for the CA where you have bought code signing ID.

Do not use a shared network folder to store this executables. The programs do not run correctly on a shared network folder.

4.5.3 Create a Signature

It is recommended to put all required tools in one folder and to add this location to the PATH variable. Each time the .inf or .sys file is modified the signature must be created again.

Create separate folders for the 32 bit and the 64 bit .inf and .sys files. The inf2cat program searches all files in the folder and cannot handle the 32 bit files in 64 bit mode correctly. To create the CAT file for 32 bit files execute the following command line:

```
inf2cat.exe /driver:<folder with 32 bit files>  
/os:2000,XP_X86,Server2003_X86,Vista_X86,7_X86
```

To create the cat file for the 64 bit files run:

```
inf2cat.exe /driver:<folder with 64 bit files>  
/os:XP_X64,Server2003_X64,Vista_X64,7_X64
```

Finally the signature can be created. Run the following command line:

```
signtool sign /v /n <Certificate Name>  
/ac <cross certificate with path>.cer  
/t http://timestamp.verisign.com/scripts/timestamp.dll  
<cat file with path>.cat
```

This program requires access to the Internet to get the time stamp from verisign.com. A digital ID from a different CA may require a different time stamp server. The Certificate Name can be displayed with the certificate manager. To open the certificate manager enter certmgr.msc on a command line. To show the signature of the .cat file use the property page. To verify the signature use the command line:

```
signtool verify /pa <path and name of the catalog file>.cat
```

The most common error occurs because the cross certificate is not added correctly. To check that the cross certificate is part of the signature open the file properties of the signed file -> Digital Signatures -> Details -> Advanced. The last two lines in the dialog box should contain 'unauthenticated attributes' and 'Counter Sign'. If this entry is missed you are using the wrong version of the signtool or the switch /ac was not used during signing.

4.5.4 Modified System Behavior

If the driver package has a valid signature from a vendor the system behavior is modified in the following way:

On Windows 2000 and Windows XP the property page of the device manager shows that the driver does not have a valid signature. Both operating systems cannot validate the signature because the trusted chain of the certificates is not completely stored in the operating system. Both systems accept only a signature created by Microsoft during a WHQL certification process.

On Windows Vista, Vista x64 and later systems during the driver installation a message shows the name of the vendor that has signed the driver package. The user can select whether the driver package should be installed or the installation should be aborted.

If the certificate of the vendor is stored in the Trusted Vendors section of the certificate manager and if the driver is pre-installed the driver is installed silently on Windows Vista and later systems.

5 Driver Installation and Uninstallation

This section discusses topics relating to the installation and un-installation of the CDC ECM device driver.

5.1 Driver Installation for Developers

This section describes some methods how developers can install and un-install the CDC ECM driver on a PC.

5.1.1 Installing CDC ECM Driver Manually

The manually installation of driver should be used by developers, only.

In order to install the CDC ECM driver manually you have to prepare an INF file that matches your device.

The steps required to install the driver are described below.

- Copy the .inf, .sys and .cat files that are required for the used operating system to the hard disk. Use a folder like
`c:\program files\
Do not copy the files in system folders.`

If you install the driver files direct from a removable medium the system asks you on the next installation process to insert the medium again.

- Connect your USB device to the system. After the device has been plugged in, Windows launches the New Hardware Wizard and prompts you for a device driver. On Windows Vista and later the system shows you a short notification that the driver was not installed correctly. You have to open the device manager manually and install the ECM driver.

Provide the New Hardware Wizard with the path of your installation files (e.g. cdcecm.inf and cdcecm.sys for Windows 2000/XP/Vista/7). Complete the wizard by following the instructions shown on screen. If the INF file matches your device, the driver should be installed successfully.

Note that the New Hardware Wizard shows a warning message that complains about the fact that the driver is not certified and digitally signed. You may ignore this warning and continue with driver installation. The CDCECM driver is not signed because it is not an end-user product. When the driver is integrated into such a product, it is possible to get a certification and a digital signature from the Windows Hardware Quality Labs (WHQL). On Windows Vista and later a digital signature of the vendor improves this behavior.

- If the operating system contains a driver that is suitable for your device, the system does not launch the New Hardware Wizard after the device is plugged in. Instead of that a system-provided device driver will be installed silently. The operating system does not ask for a driver because it finds a matching entry for the device in its internal INF file data base.

You have to use the Device Manager to install the driver for a device for which a driver is already running. To start the Device Manager, right-click on the "My Computer" icon and

choose Properties. In the Device Manager, right-click on your device and choose Properties. On the property page that pops up choose Driver and click the button labeled "Update Driver". The Upgrade Device Driver Wizard is started which is similar to the New Hardware Wizard described above. Provide the wizard with the location of your installation files and complete driver installation by following the instructions shown on screen. To make sure the system installs the correct driver don't search for the best driver. Select it from a list and use the button "Have Disk" to browse to the file location.

- After the driver installation has been successfully completed your device should be shown in the Device Manager in the section Network Adapters. You may use the Properties dialog box of that entry to verify that the driver is installed and running.
- To verify that the driver is working properly with your device, you can check the adapter with the command line tool `ipconfig`. Depending on the device capabilities the adapter gets an IP address from DHCP or Auto IP. You can assign a static IP address in the network dialog. You can try to use the command line tool `ping` to get contact to your device. Consider fire wall settings on the PC.

5.1.2 Uninstalling CDC ECM manually

The manually un-installation of driver should be used by developers, only. If an important system driver is removed the system can become unusable.

To uninstall the device driver for a given device, use the Device Manager. The Device Manager can be accessed by right-clicking the "My Computer" icon on the desktop, choosing "Properties" from the context menu and then opening the Device Manager window. Within the Device Manager window, double-click on the entry for the device and choose the property page labeled "Driver". There are two options to uninstall the device driver:

- Remove the selected device from the system by clicking the button "Uninstall". The operating system will re-install a driver the next time the device is connected or the system is rebooted.
- Install a new driver for the selected device by clicking the button "Update Driver". The operating system launches the Upgrade Device Driver Wizard which searches for driver files or lets you select a driver.

In order to avoid automatic and silent re-installation of the driver by the operating system, it is necessary to remove the driver installation from the system.

On Windows Vista and higher the complete driver package with `.sys`, `.inf` and `.cat` files is stored in the driver store. To remove it from the driver store the dialog that is shown during the uninstallation allows to delete the drivers in the store. On Windows 2000 and Windows XP the `.inf` file must be removed manually.

During driver installation, Windows 2000 and Windows XP store a copy of the INF file in its internal INF file data base located in `%WINDIR%\INF\`. The name of the INF file is changed before it is stored in the database. On Windows 2000 and Windows XP the INF file is stored as `oemX.inf`, where X is a decimal number.

The best way to find the correct INF file is to do a search for some significant string in all the INF files in the directory %WINDIR%\INF\ and its subdirectories. Note that on Windows 2000/XP, by default the %WINDIR%\INF\ directory has the attribute Hidden. Therefore, by default the directory is not shown in Windows Explorer.

On Windows Vista and higher the .cat file is stored on an additional place. If you install the same driver a second time, the system behavior may be different, because the system knows the driver and does not warn the user for a not certified driver. Furthermore the digital signature can be stored in the certificate manager under Trusted Publishers. This happens if the user selects "always trust this vendor" during the installation process. If the certificate is stored there all digitally signed drivers can be installed in the same way as with a WHQL signature.

Once you have located the INF file, delete it. This will prevent Windows from reinstalling the driver. Instead, the New Hardware Wizard will be launched and you will be asked for a driver.

5.2 Installing CDC ECM driver for End Users

This section describes ways how the driver should be installed on a PC by the end user of the product.

5.2.1 Installing CDC ECM driver with the PnP Driver Installer

Thesycon provides a PnP Driver Installer Package that can be used to install kernel mode drivers in a convenient and reliable way. This installer is not part of this package. It can be downloaded separately under the following link: <http://www.thesycon.de/pnpinstaller>.

The installation program can be run in an interactive mode with graphical user interface or it can be run in a command line mode. The command line mode is designed to integrate the driver installer into other installation programs. The GUI mode guides the user through the installation. It supports different languages.

The driver installer package can be customized. A detailed description of the customization options is part of the reference documentation.

The PnP Driver Installer Package can handle the driver installation and un-installation in different situations:

- During the first time installation the driver is pre-installed in the system. In this step the user needs administrator privileges. When the driver is certified the pre-installation of the driver is performed silent. This means the hardware wizard is not launched and the system does not show a warning box for not certified software. When the device is connected to the PC during the installation the correct driver software is installed immediately. When a device is connected later to the PC the certified driver is installed silent. At the point of time where the device is connected to the PC no administrator privileges are required.
- The installer can perform a driver update. Old drivers and driver instances are removed from the system regardless whether the devices are connected or not. Exactly the driver version from the installer package is installed. This way of driver updates enables the upgrade to higher driver versions as well as the installation of older versions. The driver installation behaves in the same way as the first time installation.

- The installer can remove the driver software for a PnP device. This step can be performed by calling the installation program with a command line option or by using the appropriate option in the Windows control panel that is created during the driver installation. When the driver is removed all device nodes are uninstalled and the pre-installed drivers are removed. The installer makes sure that all drivers that are matching the USB VID and PID are removed from the system. The result is a system that behaves in the same way as when the driver software was never installed. The device nodes are left uninstalled. When a device is connected to the PC in this state the Found New Hardware wizard is launched.

The PnP Driver Installer Package supports all current Windows systems including 32 and 64 bit versions. The Thesycon team provides support and warranty for the product. A comprehensive documentation is part of the demo package available at <http://www.thesycon.de/pnpinstaller>.

5.2.2 Installing the Driver with DIFx

The DIFx software has some limitations that are explained in this section. For that reason the installation with this tools is only the second choice and not recommended by Thesycon.

DIFx is the Driver Install Frameworks provided by Microsoft. This software component can be redistributed. DIFx consists of the driver framework for applications, the Driver Install Frameworks API and the Driver Package Installer. This section introduces the Driver Package Installer, only. For details to the DIFx framework please refer to the Microsoft documentation.

The Driver Package Installer is an executable program. It is available for x86 and x64 in two different executables with the same name 'DPInst.exe'.

The Driver Package Installer can be run in silent mode or with user interface. It can be used to uninstall a driver package and it can be configured with command line parameters.

The installer has some drawbacks:

- Depending on the device connection state it updates the driver for connected devices and it pre-installs the driver for not connected devices. If the device is connected later the system performs the normal driver rating and may select a different driver. If the driver is updated with the /f (force) flag other drivers are overwritten for connected devices. The result of the installation depends from the fact whether the device was connected or not during the installation process. It is not sure that the driver from the installation package is installed for the device.
- After uninstalling a driver the installer activates an other driver that is pre-installed for the device or it leaves the node uninstalled. This can be an older version of the driver or a demo package. The state after uninstalling depends from other drivers that may be installed on the system.

6 Debug Support

6.1 Event Log Entries

If the driver detects a major problem during the startup process it creates an entry in the event log of the system. The event log can be opened with the context menu on 'My Computer' -> Manage -> Event Viewer -> System. The entries are created in the category error.

Common problems are:

6.1.1 Clear Feature Endpoint Halt

The device cannot handle this request. Solution: Set the flag `SendClearEpHaltOnStart` to 0.

6.1.2 Demo is Expired

The demo version has an internal clock. If the demo period has been expired the driver fails all operations. To re-enable the driver reboot the PC. This does not occur with the release version.

6.2 Enable Debug Traces

The licensed version of the driver package contains a folder `drv_chk` with the debug version of the driver. This folder is not part of the demo version. The debug version of the driver can generate text messages on a kernel debugger or a similar application to view the kernel output. These messages can help to analyze problems.

To enable the debug traces follow these steps.

- Install the driver with the normal setup.
- Copy the debug version of the driver to the folder `%SystemRoot%\system32\drivers`. If the driver has been renamed the debug version must be renamed in the same way.
- Reboot the PC to make sure the new driver is loaded. Connect your device.
- Open the registry editor. On Windows 2000/XP/Vista/W7 the following path must be opened:

```
\HKEY_LOCAL_MACHINE\System\CurrentControlSet\services\  
YOUR_SERVICE_NAME\.
```

`YOUR_SERVICE_NAME` is the name of the service name defined in the `.INF` file. It is typically the name of your driver.

Edit the `DWORD` value key `TraceMask`. The messages are grouped in special topics. Each topic can be enabled with a bit in the debug mask. To enable the messages on bit 5 the `TraceMask` must be set to `0x00000020`. The `TraceMask` contains the or'ed value of all active message bits.

- On Windows Vista and later an additional registry key must be created: `\HKLM\system\CurrentControlSet\Control\Session Manager\Debug Print Filter`. In this key a DWORD value with name `DEFAULT` and the value `0xf` must be created. This enables the system debug print filter.
- Disconnect all devices or reboot the PC to make sure the driver is loaded again. The driver reads the registry key `TraceMask` if it is started.
- Start a kernel debugger like WinDbg or an application like DebugView to receive the kernel traces. DebugView is a free software that can be downloaded on the Microsoft web page. Connect your device.
- If your problem ends up in a blue screen of death or you see an unexpected re-boot of the PC please change the following settings: System Properties -> Advanced -> Startup and Recovery -> Settings -> Write Debugging Information to "Kernel Memory Dump". Set the registry key `TraceLogSizeKB` in the same location as the `TraceMask` to 128. This includes the last 128kb traces to the memory dump. Reproduce the BSOD again. Transfer the memory dump (typically `%SystemRoot%\MEMORY.DMP`) to Thesycon for analysis.

The following table summarizes the meaning of the debug bits.

Table 1: Trace Mask Bit Content

Bit	Content
0	Fatal errors
1	Warnings
2	Information
3	Extended information
4	OID based requests
8	Tx data path handling
12	Data dump
16	Rx data path handling

If you submit a problem description to Thesycon please include the following:

- operating system and Service Pack
- USB host controller (EHC, UHC, OHC) and usage of external USB hubs
- Driver version and driver build (release/debug/demo) that has been used
- used `TraceMask` if kernel Traces have been recorded

- The actions you have performed to cause the problem

Do not send memory dumps with e-mail. We can provide a FTP account.

7 Related Documents

References

- [1] Universal Serial Bus Specification 1.1,
<http://www.usb.org>
- [2] Universal Serial Bus Specification 2.0,
<http://www.usb.org>
- [3] USB device class specifications (Audio, CDC, ECM etc.),
<http://www.usb.org>
- [4] Microsoft Developer Network (MSDN) Library,
<http://msdn.microsoft.com/library/>
- [5] Windows Driver Development Kit,
<http://msdn.microsoft.com/library/>
- [6] Windows Platform SDK,
<http://msdn.microsoft.com/library/>

