



Thunderbolt™

Software Development Kit Guide

Version 3.2

Document Release Date: November 3, 2015

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Legal Notices and Disclaimers

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order. Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm>

Intel, Thunderbolt, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2014-2015, Intel Corporation. All rights reserved.

Table of Contents

Version History	1
1 Introduction	2
2 Prerequisites	2
3 Thunderbolt WMI API Reference	3
3.1 Namespace	3
3.2 WS-I Compatibility	3
3.3 Classes	4
3.3.1 SdkTbtController	4
3.3.2 SdkTbtDevice	4
3.4 Methods	6
3.4.1 UpdateFirmware	6
3.4.2 GetCurrentNvmVersion	6
3.4.3 GetCurrentPDVersion	6
3.4.4 ReadFirmware	7
3.4.5 Return Codes	7
3.4.6 Remarks	8
4 Thunderbolt FW Update Sample Code	9
4.1 FwUpdateAPI Project	9
4.1.1 SdkTbtBase	9
4.1.1.1 UpdateFirmware	9
4.1.1.2 GetCurrentNvmVersion	9
4.1.1.3 GetCurrentPdVersion	10
4.1.1.4 ReadFirmware	10
4.1.1.5 ValidateImage	10
4.1.1.6 UpdateFirmwareFromFile	10
4.1.2 SdkTbtController	11
4.1.2.1 GetControllersFromWmi	11
4.1.3 SdkTbtDevice	11
4.1.3.1 GetDevicesFromWmi	11
4.1.4 TbtException, TbtStatus and Error Handling	11

4.1.4.1	TbtException.....	11
4.1.4.2	TbtStatus	11
4.1.5	Utilities	13
4.1.5.1	GetImageNVMVersion	13
4.2	FwUpdateTool Project	14
4.3	FwUpdateCmd Project	15
4.3.1	Remarks	16

Version History

Version	Date	Comments
3.2	November 3, 2015	Updated document to support Thunderbolt'15 SW3 release (package version 15.3.39.x) - SDK and samples updated to support exposure of power delivery (PD) firmware version
3.1	August 18, 2015	Added reference to the Host Controller Force Power sample
3.0	August 9, 2015	Updated document to support Thunderbolt'15 SW2 Prime release (package version 15.2.35.x) - SDK and samples updated to support firmware update on device (for L6000 controller series only) - Modified samples with better error handling
2.1	June 25, 2015	Updated document to support Thunderbolt'15 SW2 release (package version 15.2.32.x) - Modified API to support L6000 controller series - Added command line sample - Added return code documentation
2.0	June 15, 2014	First version of Thunderbolt™ Software Development Kit for firmware update Aligned with Thunderbolt'14 SW2.5 and L5000 controller series

1 Introduction

This guide describes how to use the Thunderbolt™ Software Development Kit (SDK).

The purpose of this guide is to provide information to OEMs that want to control the firmware update process of Thunderbolt controllers. This guide includes documentation only for the classes and methods that can be used to update the firmware of Thunderbolt controllers.

Note:

The SDK includes two sample projects, FwUpdateTool (GUI) and FwUpdateCmd (CLI), located in the Samples folder (in FW_Update_Tool.zip file). The samples show how to use the methods in this SDK to update the Thunderbolt firmware. The sample projects are written in C# using Microsoft* Visual Studio 2013.

2 Prerequisites

These are the prerequisites to use this SDK:

- The Thunderbolt software that comes with the SDK must be installed on the operating system
- For host controller firmware update, at least one device must be connected to a Thunderbolt port (plugging in a device powers up the Thunderbolt controller and loads the required device driver) or the Thunderbolt host controller should be powered ON through BIOS interface. For device firmware update, the device must be connected to a supporting host controller.

Note:

The SDK includes samples in the ControllerForcePower folder located in the Samples folder (in ControllerForcePower.zip). The samples show how to use the methods to force power the controller, through the Windows* WMI ACPI. The samples were written in VBScript and C# using Microsoft* Visual Studio 2013.

- Supported operating systems:
 - Windows* 7 32-bit and 64-bit (with .NET 4.5)
 - Windows* 8.1 64-bit
 - Windows* 10 64-bit
- Supported Thunderbolt controllers:
 - On host: L5000 Series and higher
 - On device: L6000 Series device connected to L6000 Series host controller

3 Thunderbolt WMI API Reference

The Thunderbolt API enables applications to interface with Thunderbolt software using Windows* Management Instrumentation (WMI) and Managed Object Format (MOF) files.

For more information, see:

- [Namespace](#)
- [WS-I Compatibility](#)
- [Classes](#)
- [Methods](#)

Note:

This guide includes documentation only for the classes and methods that can be used to update the firmware of Thunderbolt controller.

3.1 Namespace

The WMI classes and methods are registered in this namespace:

Root\Intel\Thunderbolt

3.2 WS-I Compatibility

The Thunderbolt MOF file is not fully compliant with the Web Service Interoperability (WS-I) requirements as defined by the WS-I Organization. While the class definitions were not modified, implementation specific qualifiers (such as "Local" and "Dynamic") were appended to some of the class and method declarations.

3.3 Classes

This section describes the Thunderbolt API classes that contain the firmware related methods.

Both classes support the same methods as is described in a later section. This section describes the class properties of each class.

3.3.1 SdkTbtController

This class contains methods that you can use to control the firmware update process of a specific Thunderbolt host controller. When host controller is in safe mode, only UpdateFirmware method is available.

Fields

Type	Name	Value or Description
String	ControllerId	The ID of the host controller
Boolean	IsInSafeMode	TRUE if the host controller is in safe mode
UInt8	NVMVersion	The NVM version of the current firmware
String	PDVersion	The power delivery firmware version

Methods

- [UpdateFirmware](#)
- [GetCurrentNvmVersion](#)
- [GetCurrentPDVersion](#)
- [ReadFirmware](#)

3.3.2 SdkTbtDevice

This class contains methods that you can use to control the firmware update process of a specific Thunderbolt device controller.

Fields

Type	Name	Value or Description
String	UUID	The UUID of the device controller
String	ControllerId	The ID of the host controller the device is connected to
UInt32	PortNum	0-based index of the port in the host controller the device is connected to
UInt32	PositionInChain	1-based index of the device position in the port the device is connected to

Type	Name	Value or Description
String	VendorName	Device vendor name
String	ModelName	Device model name
UInt32	VendorId	Device vendor ID (Note: the actual value is bounded to be UInt16)
UInt32	ModelId	Device model ID (Note: the actual value is bounded to be UInt16)
Boolean	Updatable	True if the device is updatable If the host controller doesn't support device FW update, the device appears like not updatable even if it supports FW update

Methods

- [UpdateFirmware](#)
- [GetCurrentNvmVersion](#)
- [GetCurrentPDVersion](#)
- [ReadFirmware](#)

3.4 Methods

This section describes the methods available in the SDK for both SdkTbtController and SdkTbtDevice classes.

3.4.1 UpdateFirmware

This method starts the update process of the Thunderbolt firmware. For a host controller that is in safe mode, this is the only available method.

Syntax

```
uint32 UpdateFirmware(uint32 bufferSize, uint8 buffer[])
```

Parameters

Name	Type	Value or Description
bufferSize	Input	The firmware binary image size
buffer	Input	The firmware binary image

Return Values

- See [return codes table](#)

3.4.2 GetCurrentNvmVersion

This method returns the Non-Volatile Memory (NVM) revision number of the Thunderbolt firmware. This method isn't available for host controllers in safe mode.

Syntax

```
uint32 GetCurrentNvmVersion(uint32 nvmVersion)
```

Parameters

Name	Type	Value or Description
nvmVersion	Output	The current NVM version of the firmware

Return Values

- See [return codes table](#)

3.4.3 GetCurrentPDVersion

This method returns the power delivery (PD) firmware version. This method isn't available for host controllers in safe mode.

Syntax

```
uint32 GetCurrentPDVersion(String pdVersion)
```

Parameters

Name	Type	Value or Description
pdVersion	Output	The current PD firmware version

Return Values

- See [return codes table](#)

3.4.4 ReadFirmware

This method reads a block of data from the Thunderbolt firmware. This method isn't available for host controllers in safe mode.

Syntax

```
uint32 ReadFirmware(uint32 offset, uint32 length, uint8 data[])
```

Parameters

Name	Type	Value or Description
offset	Input	Offset of the first byte to read. For current version: Must be DWORD-aligned.
length	Input	Length in bytes of the requested data. For current version: field is ignored, length will always be 4 (one DWORD).
data	Output	Buffer that will be filled with the requested data from firmware. For current version: buffer must be one DWORD.

Return Values

- See [return codes table](#)

3.4.5 Return Codes

The return codes of the SDK API are as follows:

Value	Source	Description
0x00	General	Success
0x01	Firmware	Authentication failed

Value	Source	Description
0x02	Firmware	Access to restricted area
0x03	Firmware	General error
0x04	Firmware	Authentication in progress
0x05	Firmware	No key for the specified UID
0x06	Firmware	Authentication key failed
0x07	Firmware	Authentication bonded UUID failed
0x08	Firmware	Unused
0x09	Firmware	Host controller in safe mode
0x100	Service	Firmware response timed out
0x101	Service	Invalid image size
0x102	Service	Internal error
0x103	Service	Power cycle failed
0x104	Service	Operation isn't available when the host controller is in safe-mode
0x105	Service	Platform or controller is not supported
0x106	Service	Invalid argument
0x107	Service	Device is not supported
0x108	Service	Host controller is not supported
0x109	Service	SDK is in use (when trying to run more than one SDK command concurrently)

3.4.6 Remarks

After updating the firmware of a host controller or device controller, it can take up to 10 seconds for all devices to be reconnected. A script that tries to update the firmware of more than one device in a row must wait between the commands and then query the device list again to make sure the devices have been reconnected.

4 Thunderbolt FW Update Sample Code

This SDK includes some code to show how to implement the firmware update flow using the Thunderbolt API. The sample code is composed of an API wrapper and two sample applications (GUI based and CLI based) that use this API to perform firmware update. It can be used as reference when building other firmware update applications. The API wrapper and samples are written in C# using Visual Studio 2013 with .Net 4.5.

Note:

Prerequisites to using the application compiled from the sample code are the same as the ones mentioned [above](#).

4.1 FwUpdateAPI Project

This project includes wrappers for WMI API described above, some utilities and most of the logic used by both the GUI and CLI samples. Only the main parts are documented here. For more information please refer to the source code and the additional documentation there.

Please note: All the functions throw exceptions in case of an error. Applications are responsible for catching the exception and displaying a reasonable error message to the user, if applied. CLI sample ([see below](#)) can be used as a reference for such handling.

4.1.1 SdkTbtBase

This class is a base class for the interfacing with both host controllers and devices.

This, with the derived classes (SdkTbtController and SdkTbtDevice), is the main interface of this API module for applications to use.

The main role of this class is to tie the interface to WMI but it also includes the interface for validating the compatibility of the new firmware image with the current controller.

Four of the functions in this class (almost) mirror the [Methods in the WMI class](#).

4.1.1.1 UpdateFirmware

```
public void UpdateFirmware(uint bufferSize, byte[] buffer)
```

Wrapper for [UpdateFirmware](#) WMI method. The parameters are the same, but this function throws an exception on error.

4.1.1.2 GetCurrentNvmVersion

```
public UInt32 GetCurrentNvmVersion()
```

Wrapper for [GetCurrentNvmVersion](#) WMI method. This wrapper returns the current NVM version by return value instead of output parameter. Throws an exception on error.

4.1.1.3 GetCurrentPdVersion

```
public string GetCurrentPdVersion()
```

Wrapper for [GetCurrentPDVersion](#) WMI method. This wrapper returns the current PD firmware version by return value instead of output parameter. Throws an exception on error.

4.1.1.4 ReadFirmware

```
public byte[] ReadFirmware(UInt32 offset, UInt32 length)
```

Wrapper for [ReadFirmware](#) WMI method.

In contrast to the limitations mentioned in the WMI method documentation above, this wrapper function allows reading blocks of data from any offset and of any length. It may call the WMI method multiple times as needed and then extract the exact requested data.

This wrapper returns the data by return value instead of out parameter. Throws an exception on error.

4.1.1.5 ValidateImage

```
public abstract void ValidateImage(string path)
```

This function validates that a given image (from the binary file in the given path) is valid as it relates to the existing image, existing hardware (controller), and the available area on the chip. This function uses a table of properties, which are compared between new image and existing one, and these match the properties that are described in the NVM release notes. We recommend comparing all these properties before updating the NVM image.

The function is implemented by the derived classes, `SdkTbtController` and `SdkTbtDevice`, and, in turn, these implementations use the `ImageValidator` class hierarchy.

4.1.1.6 UpdateFirmwareFromFile

```
public void UpdateFirmware(string filename)
```

This function is a convenient utility for updating the firmware from a file, given the path to the file as an argument. It uses the [UpdateFirmware](#) method and throws an exception on error.

4.1.2 SdkTbtController

This class is a wrapper around SdkTbtController WMI class and exposes the same properties and methods as described above. Please note the differences in the methods interface as described in SdkTbtBase class documentation above.

It also includes the following static function for getting the available class instances.

4.1.2.1 GetControllersFromWmi

```
public static Dictionary<String,SdkTbtController> GetControllersFromWmi()
```

This function enumerates all host controller instances that can be detected. All firmware update operations are performed on these instances.

4.1.3 SdkTbtDevice

This class is a wrapper around SdkTbtDevice WMI class and exposes the same properties and methods as described above. Please note the differences in the methods interface as described in SdkTbtBase class documentation above. Another difference is that it exposes the VendorID and ModelID properties as UInt16.

It also includes the following static function for getting the available class instances.

4.1.3.1 GetDevicesFromWmi

```
public static Dictionary<String,SdkTbtDevice> GetDevicesFromWmi()
```

This function enumerates all device controller instances that can be detected. All firmware update operations are performed on these instances.

4.1.4 TbtException, TbtStatus and Error Handling

The file Exceptions.cs includes the tools this project and the samples use for error reporting.

4.1.4.1 TbtException

All functions in this project use this class for exceptions.

Application shall expect also exceptions that come from WMI (or the COM layer laying underneath WMI), which aren't of TbtException type, in addition to SDK exceptions.

4.1.4.2 TbtStatus

An enum that includes definition for the error codes described above ([Return Codes](#)) and additional error codes as follows:

Value	Source	Description
0x200	SDK	General error; used by sample applications for errors that are originated from WMI or COM
0x201	SDK	Internal error; used for notifying about SDK internal coding error
0x202	SDK	No command supplied; used by CLI sample when it runs without any command argument
0x203	SDK	Command not found; used by CLI sample
0x204	SDK	Argument count mismatch; used by CLI sample
0x205	SDK	Invalid host controller ID supplied
0x206	SDK	Invalid device controller UUID supplied
0x207	SDK	No file found in the supplied path for firmware image file
0x208	SDK	Service not found
0x209	SDK	Load host controllers failed
0x20A	SDK	Load devices failed
0x20B	SDK	No host controller found in system
0x20C	SDK	No device found in system
0x20D	SDK	Operation isn't available when the host controller is in safe-mode
0x20E	SDK	The device is connected to a host controller that doesn't support device firmware update
0x20F	SDK	This device controller doesn't support device firmware update
0x210	SDK	L4000 series controllers aren't supported by this SDK
0x211	SDK	Host/device controller presents an unknown chip
0x212	SDK	The supplied firmware image file is invalid (damaged file)
0x213	SDK	The supplied firmware image file failed validation (incompatible with the host/device controller)
0x214	SDK	The supplied firmware image file is for another hardware generation (incompatible with the host/device controller)
0x215	SDK	The supplied firmware image file is for different port count (incompatible with the host/device controller)
0x216	SDK	The supplied firmware image file can't fit into chip size (incompatible with the host/device controller)
0x217	SDK	Trying to update device controller with a firmware image file intended for host controller
0x218	SDK	Trying to update host controller with a firmware image file intended for device controller

Value	Source	Description
0x219	SDK	Mismatch between the supplied firmware image file and the host/device controller with regarding to PD firmware existence (one has it and one doesn't)
0x21A	SDK	No DROM section found in the supplied firmware image file
0x21B	SDK	Mismatch between the supplied firmware image file and the host/device controller with regarding to DROM section existence (one has it and one doesn't)
0x21C	SDK	The supplied firmware image file is for products of a different vendor than the host/device controller's vendor
0x21D	SDK	The supplied firmware image file is for a different product model than the host/device controller

4.1.5 Utilities

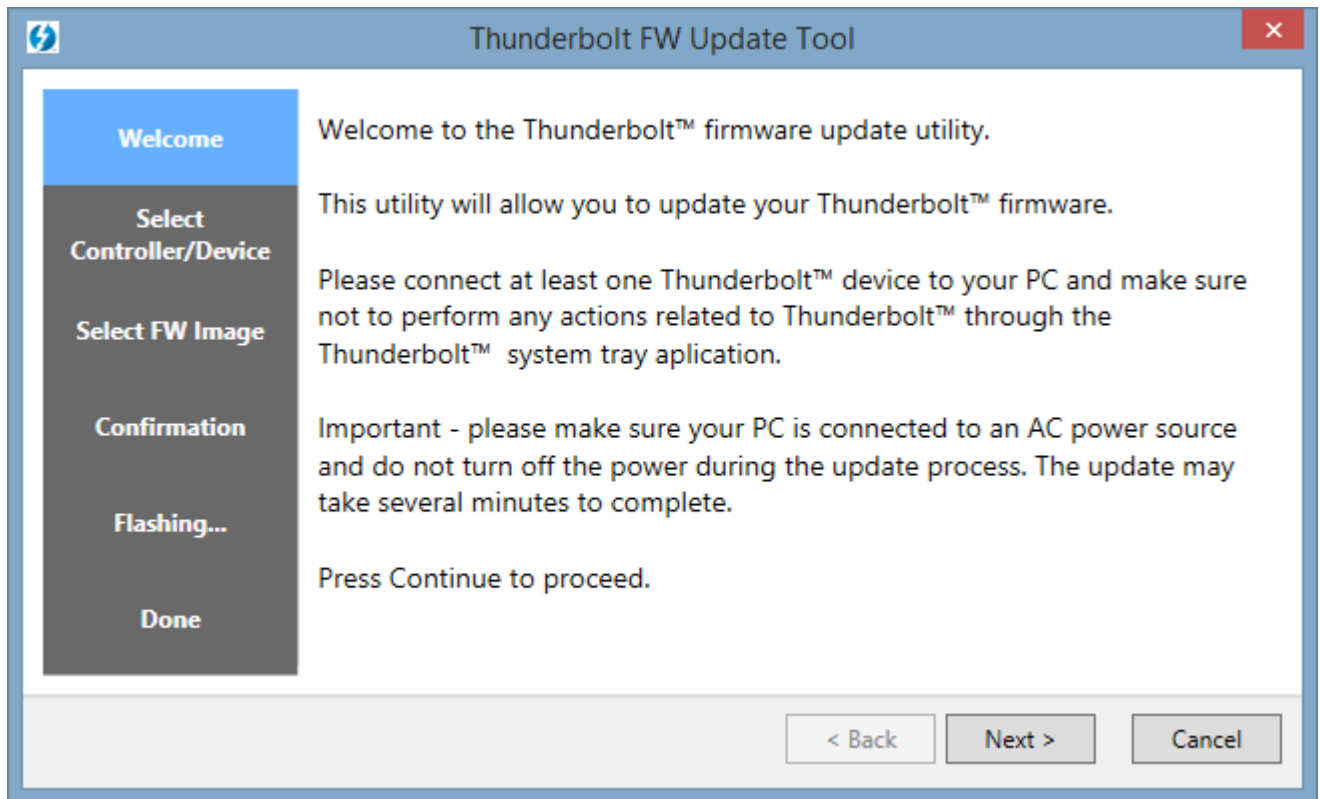
This class includes some utility functions for application developer's convenience.

4.1.5.1 GetImageNVMVersion

```
public UInt32 GetImageNVMVersion(string path)
```

Gets the NVM version from the new image (from the binary file in the given path). Used in this sample to display the new image, and allow the user to compare with the existing image version.

4.2 FwUpdateTool Project

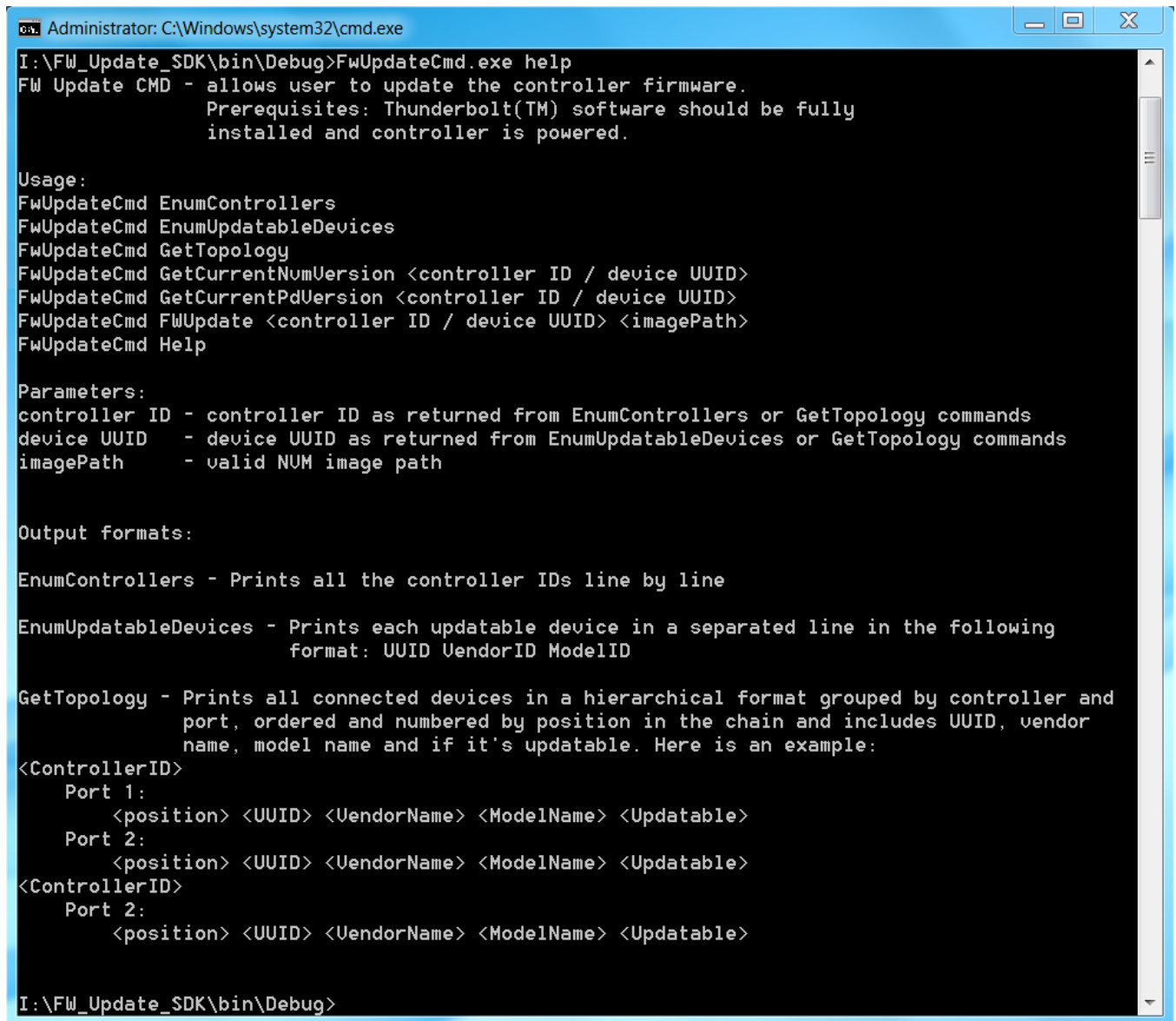


Most of the code in this sample code is UI-related and is not covered by this document.

This sample application is designed for a step by step process to perform the firmware update flow:

1. Select the Thunderbolt host controller or device to be updated
2. Select the new firmware image to be applied
3. Check compatibility between new image and selected Thunderbolt controller configuration
4. Start firmware update process

4.3 FwUpdateCmd Project



```
Administrator: C:\Windows\system32\cmd.exe
I:\FW_Update_SDK\bin\Debug>FwUpdateCmd.exe help
FW Update CMD - allows user to update the controller firmware.
                  Prerequisites: Thunderbolt(TM) software should be fully
                  installed and controller is powered.

Usage:
FwUpdateCmd EnumControllers
FwUpdateCmd EnumUpdatableDevices
FwUpdateCmd GetTopology
FwUpdateCmd GetCurrentNumVersion <controller ID / device UUID>
FwUpdateCmd GetCurrentPdVersion <controller ID / device UUID>
FwUpdateCmd FWUpdate <controller ID / device UUID> <imagePath>
FwUpdateCmd Help

Parameters:
controller ID - controller ID as returned from EnumControllers or GetTopology commands
device UUID   - device UUID as returned from EnumUpdatableDevices or GetTopology commands
imagePath     - valid NUM image path

Output formats:

EnumControllers - Prints all the controller IDs line by line

EnumUpdatableDevices - Prints each updatable device in a separated line in the following
                      format: UUID VendorID ModelID

GetTopology - Prints all connected devices in a hierarchical format grouped by controller and
              port, ordered and numbered by position in the chain and includes UUID, vendor
              name, model name and if it's updatable. Here is an example:
<ControllerID>
  Port 1:
    <position> <UUID> <VendorName> <ModelName> <Updatable>
  Port 2:
    <position> <UUID> <VendorName> <ModelName> <Updatable>
<ControllerID>
  Port 2:
    <position> <UUID> <VendorName> <ModelName> <Updatable>

I:\FW_Update_SDK\bin\Debug>
```

This sample application is command line based and was designed with automation in mind, so output would be easily parsable and reused as parameter of the different implemented commands.

It provides the following commands:

1. EnumControllers – Enumerate Thunderbolt host controllers
2. EnumUpdatableDevices – Enumerate Thunderbolt updatable devices
Please note: Devices that are connected to a host controller that doesn't support device FW update will not be listed here.
3. GetTopology – Show all devices in a tree-like format; only host controllers and ports that have devices connected to them will be shown

4. GetCurrentNvmVersion – Print current NMV version for a given Thunderbolt host/device controller
5. GetCurrentPdVersion – Print current PD firmware version for a given Thunderbolt host/device controller
6. FWUpdate – Perform firmware update on a given Thunderbolt host/device controller
7. Help – Print help about the available commands and output formats of the various host and device controller listing commands

Since this sample application is based on the same Thunderbolt API wrapper as the GUI based sample, it also performs the same kind of compatibility check between the new image and the selected Thunderbolt controller configuration before applying the new NVM image.

This sample uses application return value (ERRORLEVEL) to notify any error condition as [described above](#). It also prints the error code number, the enum entry name and detailed description if found.

4.3.1 Remarks

- This sample must run with Administrative privileges in order to be able to use the GetCurrentNvmVersion, GetCurrentPDVersion and FWUpdate commands.
- The controller ID format includes command-line special characters (e.g. '&'), so it must be quoted to pass it as an argument.