Building Devices with the MTP Porting Kit

May 1, 2007

Abstract

This paper describes the newest version of the MTP Porting Kit (Version 12) from Microsoft and how to build a MTP device using it. It is intended to give a brief overview of the porting kit as well as a recommended sequence to use to bring up a MTP stack on a device.

This information applies for the following operating systems:
Windows Vista™
Microsoft® Windows® XP

The current version of this paper is maintained on the Web at:
http://go.microsoft.com/fwlink/?LinkId=87961

Contents

Disclaimer

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

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

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

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

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

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

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

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

Introduction

The Media Transfer Protocol (MTP) Porting Kit contains the source code for a reference implementation of an MTP responder. A typical responder is a portable media player that plays audio files or a digital camera that takes photographs. A sophisticated responder might have additional capabilities—for example, playing podcasts or capturing video. The responder connects to an MTP initiator that transfers audio, video, and image files to and from the responder. An example of an initiator is an instance of Microsoft® Windows Media® Player running on a Microsoft Windows® computer. The responder communicates with the initiator through a USB connection or a network connection (typically, wireless).

Microsoft developed MTP in response to difficulties users had experienced in connecting their portable media players to Windows computers. When hardware vendors introduced the first portable devices, the devices from each vendor used a proprietary protocol for communicating with the computer. Each vendor supplied a driver that ran on the Windows computer to communicate with their devices. Users were unable to use these portable devices until they had located and installed the appropriate drivers for their devices.

To improve the user experience and relieve hardware vendors of the task of writing drivers, Microsoft has defined a standardized protocol for communication between portable media players and Windows computers—the Media Transfer Protocol. In place of proprietary drivers, Microsoft supplies an MTP driver to run on Windows computers.

The MTP driver communicates with all MTP-compatible devices and is effectively part of the Windows operating system. After buying an MTP-compatible portable media player, the user removes the portable device from the packaging and immediately connects the device to the Windows computer. No further setup is required.

By making their portable media players compatible with MTP, hardware vendors no longer have to define proprietary communication protocols and develop drivers to support them. However, developing the firmware to run the portable devices can still be a daunting task.

To simplify this task, Microsoft introduced the MTP Porting Kit in 2005. The initial porting kit contained the source code for the firmware to implement a basic MTP responder. This implementation of the responder supports a USB connection to the initiator and implements the core MTP operations, as defined in the MTP specification. In addition, Microsoft defined a digital rights management (DRM) extension to the MTP specification. Hardware vendors can extend the source code in either kit to support the special features of their portable devices.

In 2007, Microsoft released a modified version of the MTP Porting Kit. This version contains an updated implementation of the MTP responder software that supports both USB and MTP/IP connections to initiators. MTP/IP defines communication between and MTP responder and MTP initiator over an IP network connection. Typically, the MTP/IP connection is wireless.

To support both USB and MTP/IP connections required Microsoft to completely redesign the MTP responder software. A responder might be required to communicate with an initiator through a USB connection and with one or more additional initiators through MTP/IP connections. To manage these connections concurrently requires multithreading. An internal messaging system was devised to provide communication and synchronization between threads. In addition, the data transport code was tuned to eliminate data copying, reduce the number of data conversions, trim internal latencies, and boost throughput during long data transfers.

Microsoft has defined the MTP protocol as a client extension to the Picture Transfer Protocol (PTP) for digital still cameras. For information about PTP, see the PIMA 15740 (PTP) specification "Picture Transfer Protocol (PTP) for Digital Still Photography Devices," Version 1.0, at the PIMA 15740: 2000 Picture Transfer Protocol Web site (http://go.microsoft.com/fwlink?LinkId=26301).

Microsoft has based its definition of MTP/IP on the PTP/IP specification (CIPA-005/2005). For more information, see "Picture Transfer Protocol over TCP/IP Networks" at the Camera and Imaging Products Association Web site (http://www.cipa.jp/ptp-ip/index_e.html).

Supported Features

Version 12 of the MTP porting kit contains two significant changes from version 11, a new architecture designed from the ground up with devices in mind and support for the IP transport layer. For a list of all of the MTP operations and events supported by version 12 of the MTP porting kit refer to the DeviceSettings.xml located at %MTPPKROOT%\Windows\src\MTP\CoreHandler\DeviceSettings.xml. Object property support can be found in %MTPPKROOT%\Windows\src\MTP\PropAPI\MTPPropertyUtil.c. The porting kit does not contain a full persistent database; because of this not all of the PlaysForSure operations and properties are supported.

Architectural Overview

A new architecture was used for the porting kit. The new architecture was designed to be device friendly with an emphasis on compatibility and performance.

Compatibility

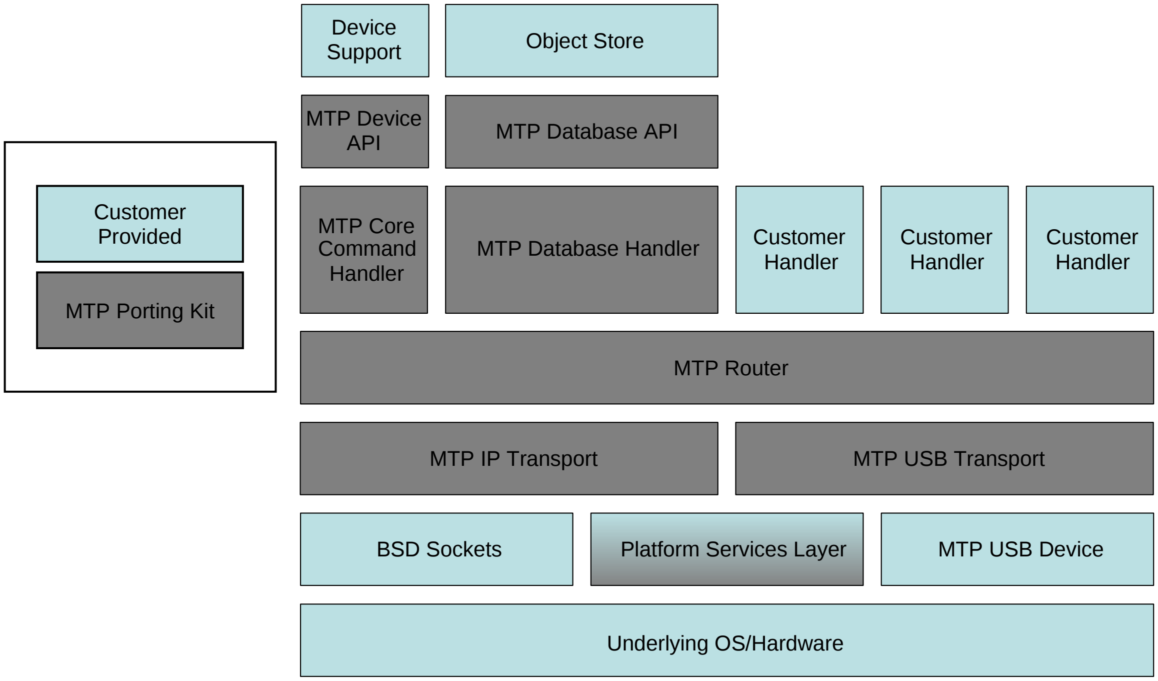
The MTP porting kit has been redesigned to be more portable. The code is divided into two main sections: code that is intended to be ported and code that should work on the majority of devices without modification. The code that does not need to be ported is referred to as common code and is located in the common directory of the porting kit.

ANSI C

The common responder code is ANSI C compliant. This code should compile under any ANSI C compatible compiler. Some of the Windows specific code is written in C++ and is not ANSI C compatible.

Component Breakdown

The new MTP porting kit architecture breaks down into components that pass and handle MTP operations. The porting kit assumes that the device is using an underlying operating system that provides a basic interface to the hardware.



- **Platform Services Layer**

The platform services layer is designed to provide a general purpose library of common functions that are used by the porting kit. The PSL layer is implemented in the common code where possible and in the device specific code for areas that are dependant on the device. For a complete list of the provided PSL functions refer to the MTP_Resp.chm help file.

- **Transport**

The transport is the primary entry point into the MTP responder. The transport section has a common section for the USB and TCP/IP transports as well as a device specific section. Packets are copied to buffers in the transport where the transport header is stripped off and the rest of the packet is sent to the router. The transport manages a pool of buffers that are used for commands, data, and responses. Each router thread has its own pool of buffers.

- **USB**

The device specific section of the USB responder contains a driver interface for the Netchip NET 2280. A USB device thread is created to communicate with the MTP router and the device-dependant driver interface. The common USB section provides the functions manage and control the buffers.

- **TCP/IP**

The TCP/IP transport is designed by using Berkley sockets. A service block and a bindrouter block make up the IP transport. The service block starts and stops the MTP/IP connection. The bindrouter thread creates the router threads that service the MTP/IP connections.

- **Router**
  The router component constructs a route from the transport to the command handler using the operation header in the packet. Each operation will receive a route for the duration of the operation. The router is implemented in common code.
- **Command Dispatcher**
  The command dispatcher reads messages from the command handler message queue and dispatches them to the command handler. At a high-level view, the command dispatcher is part of the router. The command handler is implemented in the common code.
- **Command Lookup Table**
  After receiving a new command from the initiator a new route is created by the router. A lookup table is used to determine the command handler to the requested operation.
- **Command Handler**
  It is command handler that processes the message. For a list of the command handlers in the common code, refer to MTP_Resp.chm. When sending data back to the initiator, the command handler will request a buffer from the transport.
- **Database**
  There is a database API that interfaces into the object store. The database handler is device/database specific and therefore is to be written by the customer. The object store in the porting kit is simulated with a link list and a database handler.

Custom Command Handler

Custom command handlers are available to add custom MTP extensions to the MTP porting kit. Some examples of custom command handlers that device vendors may want to implement are the DRM extensions. There is an example custom handler in the porting kit that is documented in the MTP_Resp.chm file.

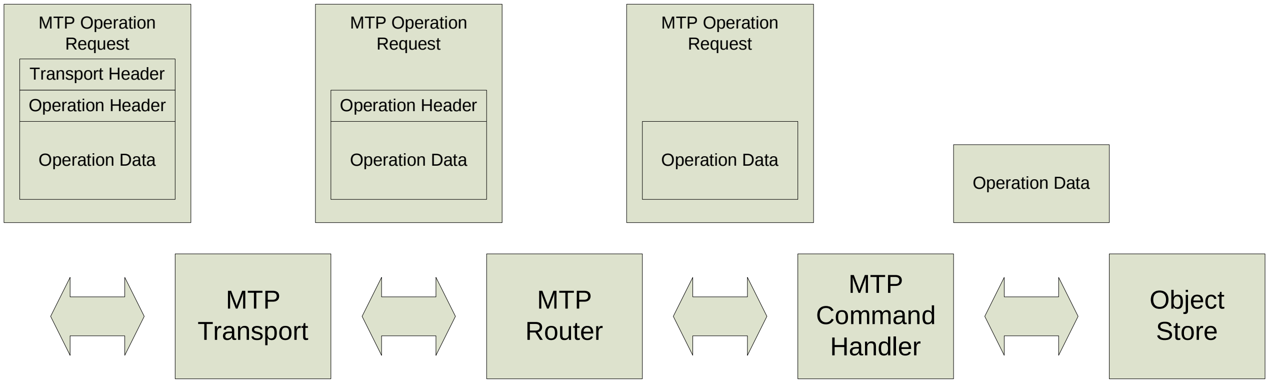
Performance

Asynchronous Model

An asynchronous model was used to take advantage of any parallelism that a processor might be capable of. A message system passes MTP messages to the different MTP components the message passing can be asynchronous. Inside the porting kit are threads associated with both common code and device-specific code. In the device-specific code the transport, dispatcher, and database all run in their own threads. The PSL functions provide basic message passing and synchronization.

Decreased Copies

Network based architecture was used to improve the performance of the data transfers. Data copies are kept to a minimum from the time the data enters the device to the time it is put onto the data store. At each stage through the architecture a pointer is passed and the pertinent information is striped from the packet.



Naming Conventions and Code Organization

The code is organized into a common section and a section that is intended to be "ported." To help assist in the porting process Platform Service Layer (PSL) functions are provided. For a full list of the PSL functions and how to use them, refer to the MTP_resp.chm help file.

Porting to Your Device

You should first read the MTP_resp.chm and PortableMediaPlayer_MTP_Guidelines.doc. These files will give you the details about the porting kit and MTP. After the porting kit is installed, use the included tools to familiarize yourself with MTP and the kit. DirectMTP is a tool that will allow you to send select MTP commands directly through the MTP driver on a PC. WPDMon will allow you to see the packets that are sent and received from the initiator. These two tools can be useful in testing and diagnosing MTP failures as you begin to bring up a device.

- When bringing up a MTP device, it is recommended you break down the work into phases.
- Phase 1 is the installation of the device and getting the device to respond to GetDeviceInfo.
- Phase 2 is adding support for browsing the device by adding support for the object store and GetObjectInfo.
- Phase 3 is object transfer support by implementing GetObject, SendObject, and DeleteObject.
- Phase 4 is property support, GetObjectPropDesc, GetObjectPropValue, and SetObjectPropValue.
- Phase 5 is the remaining miscellaneous operations and properties that the device needs to support.
- Phase 6 is to optimize the device.

While bringing up the MTP portion of your device, you can implement the device side data store and optimize the database in parallel.

The new porting kit's architecture and performance improvements should allow you to reuse a significant portion of the code on your device. This reuse of code should help increase quality as well as reduce the development time.

Resources

Media Transfer Protocol Porting Kit
http://www.microsoft.com/downloads/details.aspx?FamilyId=A2E73160-E862-4F19-BB26-C0CAFE798955&displaylang=en

PlaysForSure Test Kit
http://www.playsforsure.com/product/specifications/

PlaysForSure Help Desk
To register, send e-mail to **pfsinfo@microsoft.com** with "Help Desk Registration Request" in the subject line.
http://connect.microsoft.com/pfs