MTP Responder Development Guide

Windows Embedded CE 6.0 R3 Technical Article

Writers: Mark McLemore, Harold Drews

Technical Reviewer: Alex Bussmann, Kwan-Sub Shin, Vasu Pai

Published: December 2009

Applies To: Windows Embedded CE 6.0 R3

Abstract

Device Stage, introduced in Windows 7, helps users discover and use devices that are connected to their personal computers. MTP Responder for Windows Embedded CE 6.0 R3 adds functionality to Windows Embedded CE 6.0 R3 that helps OEMs create devices that are compatible with Device Stage. If you are creating a device that connects to a computer running Windows 7 through a USB or TCP/IP connection, MTP Responder for Windows Embedded CE 6.0 R3 gives you the necessary enhancements to build Windows 7-compatible Device Stage support into your device.

As an OEM, you can build Windows Embedded CE 6.0 R3 devices that improve the experience Windows 7 users have with their devices. For example, you can make it easier for users to find the representation of their device on the computer, you can provide users with information that improves their experience with the device, and you can present the status information about the device and the tasks that users can perform with the device, all in a consolidated view.

MTP Responder for Windows Embedded CE 6.0 R3 provides Media Transfer Protocol (MTP) functionality to support Device Stage. Although MTP Responder does not support media metadata transfer, it includes storage for MTP files and folders so that users can browse and manage files on their devices and transfer files to and from their computers.

You can modify and extend MTP Responder to support additional MTP commands and properties. By using an MTP extension, you can define new MTP operations, properties, and object formats that are not part of the MTP specification.

MTP Responder also prepares your device to meet the Windows Logo requirements for devices in the Other Portable Devices category.

Introduction

MTP Responder for Windows Embedded CE 6.0 R3 provides enhanced Media Transfer Protocol (MTP) functionality to support Windows 7 Device Stage. Device Stage offers a central location in Windows for your users to discover and use their devices. When you include Device Stage support in your device, users can get detailed information about the device, copy files to and from the device, run device-specific tasks, read a product manual, and buy accessories for the device, all in one handy location. Device Stage provides these capabilities without requiring the user to download or install software. For more information about Device Stage, see the Windows Device Experience site (http://go.microsoft.com/fwlink/?LinkId=132146).

Device Stage communicates with your device over MTP; therefore, a device must implement MTP to support Device Stage. MTP enables an application on a computer, the *MTP initiator*, to control a portable device, the *MTP responder*, and transfer digital media content and metadata between the computer and the device. The MTP initiator and the MTP responder communicate over a communications link, or *MTP transport,* typically a USB or TCP/IP connection between the computer and the device. Devices participate in Device Stage by implementing an MTP responder that can handle Device Stage requests from an MTP initiator over one or more MTP transports. The MTP responder functionality provided in this release is specifically designed to support the MTP initiator that is included in Windows 7 for Device Stage.

This guide describes the Device Stage support provided by the MTP Responder, the functionality of each MTP Responder catalog item, MTP Responder limitations, and prerequisites for including MTP Responder in your device. A detailed step-by-step Device Stage implementation section explains how to install MTP Responder, how to configure device registry settings, how to add OEM Adaptation Layer (OAL) support, how to create presentation elements, and how to test your device with Device Stage. A section on the MTP Responder design helps you learn more about the MTP Responder implementation, modify the provided source code to add customizations and extensions, and download additional software development kits and programming documentation for MTP and Device Stage development.

Device Stage Support

When you include MTP Responder for Windows Embedded CE 6.0 R3 in your device, you can choose from one of two supported levels of Device Stage presentation: *baseline* or *custom*. The baseline presentation is the simplest and most essential device presentation level, while the custom presentation offers a richer set of features for using your device.

Baseline presentation displays a device icon, some textual information about your device, and basic status information. MTP Responder provisions your device to communicate this essential information about the device and its contents to Device Stage. Using this information, MTP Responder provides the following basic Device Stage capabilities:

- **View battery status** Device Stage displays battery status information if your device reports this information.
- **View storage space** Device Stage displays the amount of storage space available on your device if your device reports this information.

- **Browse files** File browsing capability lets a user manage files and folders on a device using an Explorer window on the computer and transfer files between the device and the computer.

These basic capabilities meet the minimum functionality requirements for Windows Log certification for the Device Stage Other Portable Devices category. For more about Device Stage categories, see the section Windows Logo Certification below. Your device will automatically create a baseline presentation when you include MTP Responder catalog items in your OS design, configure registry settings with information about your device, and implement OAL support for battery and storage status.

The custom presentation extends the baseline presentation by adding enhancements such as branding logos, additional presentation images, custom tasks for managing the device, and links to provide users with access to product registration, support, manuals, applications, and accessories. To create a custom presentation, you first configure your device to support the baseline presentation, and then you create a device metadata package for installation on the user's computer to add additional end-user features. You can also implement extra functionality in your device to support custom Device Stage tasks as part of your custom presentation.

Windows Logo Certification

Windows classifies devices into categories, based on the functionality of the device. The MTP functionality in MTP Responder for Windows Embedded CE 6.0 R3 meets Windows Logo certification requirements for a new category of devices called Other Portable Devices. This category includes portable navigation devices, consumer internet devices, digital picture frames, e-book readers, portable gaming devices, and set-top boxes. The Other Portable Devices category is a member of the Portable Devices class. The Portable Devices class also contains categories for cellular phones, digital cameras, and portable media players.

Each device category has unique requirements that a device must meet to be compatible with Device Stage and qualify for a Windows Logo. The MTP components included in MTP Responder for Windows Embedded CE 6.0 R3 fulfill requirements only for the Other Portable Devices category; however, you can extend the included MTP Responder source code to meet the requirements for other Windows device categories.

For more information about Windows Logo certification and the requirements for device certification, see the Windows Logo Program (http://go.microsoft.com/fwlink/?LinkId=8772).

MTP Responder Components

MTP Responder is comprised of several major functional blocks; the *MTP Responder Stack*, *MTP Storage*, and *MTP Transports*. You select MTP Responder functionality from the Catalog Items View in Platform Builder, where several new catalog items are presented: *MTP Responder (default)*, *MTP Responder (minimal)*, *MTP USB Transport*, and *MTP IP Transport*. You cannot select MTP Storage as a separate catalog item; it is included only when you select the MTP Responder (default) catalog item. For more information about catalog items, see the section Step 2 Add Device Stage Catalog Items to Your OS Design.

MTP Responder Stack

The MTP Responder Stack provides MTP router, dispatcher, and command handler functionality to support Device Stage communication with the MTP initiator included in Windows 7. The MTP Responder Stack communicates with the user's computer through one or more MTP transports; you can configure the MTP Responder Stack to use a USB connection, a TCP/IP connection, or both. You add MTP Responder Stack functionality by adding one of the MTP Responder Stack catalog items in the Catalog Items View of Platform Builder. To support Device Stage, you must include the MTP Responder Stack in your OS design.

MTP Storage

MTP Storage adds file browsing capability over MTP by storing information about MTP objects on your device. MTP Storage is optional for Device Stage support; however, file browsing over MTP is a minimum requirement for Windows Logo certification. For more about Windows Logo certification, see Windows Logo Certification.

MTP Transports

The MTP USB Transport provides MTP connectivity between the device and the user's computer through a USB connection. The MTP IP Transport provides MTP connectivity between the device and a user's computer through a TCP/IP connection. You can use one or both of these transports in your device, but you must enable at least one transport to support Device Stage.

MTP transports can be used interchangeably. Users can copy files between the device and a Windows 7 computer over a USB connection or a TCP/IP connection. Also, users have the ability to perform the same file browsing and device management operations over a TCP/IP connection to the device as with a USB connection.

Limitations

MTP Responder for Windows Embedded CE 6.0 R3 has several limitations with respect to file formats, properties, synchronization, and storage, which are described below. However, you can modify the provided MTP Responder source code to support additional file properties and formats, add synchronization support, or replace the included MTP Storage component with your own storage implementation. For more information about these modifications, see the MTP Storage section of MTP Responder Design.

File Formats and Properties

MTP Storage supports only two object formats: *Undefined* (files) and *Association* (folders). Your device may store many different file formats (.mp3, .jpg, and so on), but MTP Storage treats them all as Undefined format. As a result, MTP Responder ignores media properties found in video and audio files.

- Properties that are specific to media files, such as Artist, Title, and Album Name cannot be stored on the device.
- Windows Media Player cannot play files transferred over MTP from the device because these files no longer have the properties that characterize media files.

Windows Media Player only recognizes devices that support media file formats and properties.

Although MTP Storage does not support media file properties, media files managed by MTP Storage contain a minimum set of properties, such as the file name and file type, that must be present for use over MTP connections.

File Synchronization

MTP Responder does not provide a mechanism for users to compare the contents of files on the device with files on a computer or update files in either location based on that comparison.

Closed Storage Implementation

Applications cannot access storage data that is created and managed by MTP Storage; only MTP Responder Stack can access files and object metadata managed by MTP Storage.

Prerequisites

To build support for Device Stage into your device by using MTP Responder for Windows Embedded CE 6.0 R3, you use the tools that you normally use for Windows Embedded CE Development. You use a few additional tools and software development kits to test your Device Stage presentation and create and test device metadata packages.

Windows Embedded CE

To support Device Stage, you must use Windows Embedded CE 6.0 R3. Earlier releases of Windows Embedded CE do not include support for Device Stage. MTP Responder for Windows Embedded CE 6.0 R3 includes an MTP responder and related functionality that is designed for servicing Windows 7 Device Stage requests.

You use the Visual Studio integrated development environment (IDE) with the Windows Embedded CE Platform Builder toolset to design, create, build, test, and debug your Windows Embedded CE-based run-time image. For more information about Windows Embedded CE development and Platform Builder, see the Platform Builder User's Guide (http://go.microsoft.com/fwlink/?LinkId=178104).

Metadata Tools

To support the Device Stage custom presentation, you must assemble the files that make up a Device Stage custom presentation into a device metadata package for installation on the user's computer (the baseline presentation does not require a device metadata package). For more information about creating a device metadata package for a custom device presentation, download the Windows Device Experience Development Kit (http://go.microsoft.com/fwlink/?LinkId=178109).

Windows 7

Windows 7 provides an MTP initiator for testing your device in addition to the environment for building and testing device metadata packages.

Development Steps

To configure your Windows Embedded CE 6.0 R3 OS design to support Device Stage, follow these steps.

1. Download and install *MTP Responder for Windows Embedded CE 6.0 R3*, available from the Windows Embedded CE 6.0 Download Center (http://go.microsoft.com/fwlink/?LinkId=179042).

2. Create a *baseline* Device Stage presentation, which displays some identifying information about your device, your device icon, and a standard set of status elements and tasks.
    a. Choose and include MTP Responder catalog items in your OS design.
    b. Create a device icon image that represents your device.
    c. Populate device registry keys with information about your device.
    d. Develop your OEM Adaptation Layer (OAL) to support Device Stage status and task elements.

3. Optionally, create a *custom* Device Stage presentation, which includes a user interface for your device that installs on Windows 7.
    a. Create a device metadata package that installs on the user's computer.
    b. Develop any custom tasks in your device to support your device metadata package.
    c. Submit your device for Windows Logo certification.
    d. Submit your device metadata package to Microsoft for signing.

When you complete steps 1 through 2 above, Device Stage generates a baseline presentation for your device automatically. When you complete steps 1 through 3 above, Device Stage displays your device with a custom presentation. Windows Logo certification and metadata signing are required only for a custom presentation.

If you create a custom presentation, you can add status displays and tasks that are specific to your device, promote your brand by adding custom graphic elements, and provide users with links to product enhancements, registration sites, support sites, and product manuals. If you don't create a user interface, then Device Stage shows the baseline presentation for your device.

The steps below and the links to related Web sites, software development kits, and additional documentation will help you prepare your device for Device Stage.

Device Stage Implementation

By using MTP Responder, you take advantage of the MTP class driver support provided in Windows, thereby reducing the need to design, develop, and support a proprietary device connectivity solution. You provide a user with a simpler connection experience because no additional software installation is necessary to communicate with Windows. To add Device Stage support to your device by using MTP Responder for Windows Embedded CE 6.0 R3, follow these steps:

Step 1 Verify System Requirements

MTP Responder for Windows Embedded CE 6.0 R3 requires the following minimum system configurations.

Windows Vista

If your computer runs on 32-bit Windows Vista, the following hardware and software is required.

Hardware Requirements

- Personal computer with a 933 MHz or faster processor (2 GHz recommended)
- 512 megabytes (MB) of RAM (1 GB recommended)
- 18 gigabytes (GB) of available space on installation drive
- 1 GB of available space on system drive
- DVD-ROM drive
- Monitor that supports 1024 x 768 screen resolution with 16-bit color

Software Requirements

- Microsoft Visual Studio 2005
- Visual Studio 2005 Service Pack 1
- Visual Studio 2005 Service Pack 1 Update for Windows Vista
- Windows Embedded CE 6.0 R3

Windows Embedded CE 6.0 is available from the Windows Embedded CE 6.0 Download Center (http://go.microsoft.com/fwlink/?LinkId=179042).

Windows XP

If your computer runs on 32-bit Windows XP Service Pack 2, the following hardware and software is required.

Hardware Requirements

- Personal computer with a 933 MHz or faster processor (2 GHz recommended)
- 512 megabytes (MB) of RAM (1 GB recommended)
- 18 gigabytes (GB) of available space on installation drive
- 1 GB of available space on system drive
- DVD-ROM drive
- Monitor that supports 1024 x 768 screen resolution with 16-bit color

Software Requirements

- Microsoft Visual Studio 2005
- Visual Studio 2005 Service Pack 1
- Windows Embedded CE 6.0 R3

Windows Embedded CE 6.0 is available from the Windows Embedded CE 6.0 Download Center (http://go.microsoft.com/fwlink/?LinkId=179042).

Step 2 Add Device Stage Catalog Items to Your OS Design

To add Device Stage support to your OS design, add one or more of the following catalog items and SYSGEN variables to your OS design. Use the MTP Responder (default) catalog item unless you intend to modify or replace the provided transport components. If you do not use MTP Responder (default), you must include MTP Responder (minimal) and at least one transport or use equivalent components of your own design.

Table 1 - SYSGENs

Catalog item	SYSGEN variable	Description
MTP Responder (default)	SYSGEN_MTP_RESPONDER	Adds all Device Stage components: MTP Responder Stack, MTP Storage, and all MTP transports. Includes the software components necessary to add Device Stage support on a CE 6.0 R3 device.
MTP Responder (minimal)	SYSGEN_MTP_RESPONDER_MIN	Adds only the MTP Responder stack; does not add MTP Storage or MTP transports.
MTP USB Transport	SYSGEN_MTP_RESPONDER_USB	Adds MTP over USB transport functionality.
MTP IP Transport	SYSGEN_MTP_RESPONDER_IP	Adds MTP over IP transport functionality.

For information about selecting catalog items, see Adding Catalog Items to an OS Design (http://go.microsoft.com/fwlink/?LinkId=179330). For information about setting SYSGEN variables, see Setting or Clearing a Sysgen Variable (http://go.microsoft.com/fwlink/?LinkId=179331).

Step 3 Configure Registry Settings in Your OS Design

To configure registry settings in your OS design to support Device Stage, modify the datasync.reg file, which can be found at this location:

%_WINCEROOT%\public\datasync\oak\files\datasync.reg

Device Information

To identify and install the device in Windows, associate the device with a metadata package on the computer, and display strings in Device Stage for a baseline presentation, you must add the device information entries listed in Table 2 to the registry key:

HKEY_LOCAL_MACHINE\Software\Microsoft\MTP\Responder

Table 2 - Device Information Registry Settings

Name	Type	Default Value	Description
DeviceModelName	REG_SZ	Windows Embedded Generic Device	The device model name that MTP uses to respond to **GetDeviceInfo** requests, for example, "Contoso GPS".
DeviceFriendlyName	REG_SZ	Windows Embedded Generic Device	The friendly name that MTP uses to respond to **GetDevicePropDesc** requests, for example "Contoso GPS".
ModelID	REG_SZ	Generated if not specified	A 128-bit GUID that associates a device with a Device Stage presentation, regardless of how the device is connected to the computer. For example, "{52620BB4-2F18-4e92-9494-A03A38719}". For more information about hardware IDs and model IDs, see the following documents in the Windows Device Experience Development Kit (http://go.microsoft.com/fwlink/?LinkID=178109): • Windows 7 Device Stage Portable Device Class Development Guide • Windows 7 Device Stage Reference Guide See also ModelID (http://go.microsoft.com/fwlink/?LinkID=179337). If you do not specify the ModelID, MTP Responder will generate one for you. See the note below for disabling ModelID.
DeviceVersion	REG_SZ	No default	The version information that MTP uses to respond to **GetDeviceInfo** requests, for example "1.2".

FunctionalID	REG_SZ	Generated if not specified	The 128-bit GUID—permanent for the life of the device—that uniquely identifies an MTP device that is connected via multiple transports, for example "{2BB4074E-E469-4d1d-A028-615D91AA4D21}". For more information about the functional ID, see the MTP Device Services Extension Specification (http://go.microsoft.com/fwlink/?LinkID=178887). If you do not specify the functional ID, MTP Responder will generate one for you at device boot time. See the note below for disabling FunctionalID.
ContainerID	REG_SZ	Generated if not specified	An identifier that informs Windows 7 that multiple functional device instances actually originate from the same physical device. For more about container IDs, see Multifunction Device Support and Device Container Groupings in Windows 7 (http://go.microsoft.com/fwlink/?LinkID=158386). If you do not specify the ContainerID, MTP Responder will generate one for you at device boot time.

Note Device Stage displays your device as a "Composite (Multi-Transport) Device" even if the device has only one MTP transport configured. Because MTP Responder always reports a functional ID to the MTP initiator on the user's computer (even if you do not configure FunctionalID in the registry settings above), Device Stage assumes that your device supports MTP connections over more than one transport. For more about multi-transport devices, see Multi-Transport Devices in Windows 7 (http://go.microsoft.com/fwlink/?LinkID=179542).

To disable the FunctionalID or ModelID settings, modify the devicesettings.xml file, which can be found at this location:

%_WINCEROOT%\public\datasync\oak\files\devicesettings.xml

In <DevicePropertiesSupported>, find the declarations for the FunctionalID and ModelID properties.

```
<DevicePropertiesSupported>
    <Base>0x5001</Base> <!-- BATTERYLEVEL -->
    <Base>0xD301</Base> <!-- FUNCTIONID -->
    <Base>0xD302</Base> <!-- MODELID -->
    <Base>0xD401</Base> <!-- SYNCHRONIZATIONPARTNER -->
    <Base>0xD402</Base> <!-- DEVICEFRIENDLYNAME -->
    <Base>0xD405</Base> <!-- DEVICEICON -->
</DevicePropertiesSupported>
```

When you remove the **FUNCTIONID** element shown above, MTP Responder will not report a functional ID to the MTP initiator, even if you specify the FunctionalID in the registry. When you remove the **MODELID** element shown above, MTP Responder will not report a model ID to the MTP initiator, even if you specify the ModelID in the registry.

Transport Information

If you are using the MTP USB transport, you must set the product ID and vendor ID (listed in Table 3) in the registry key:

HKEY_LOCAL_MACHINE\Drivers\USB\FunctionDrivers\MTPUSBFn

Table 3 - USB Transport Registry Settings

Name	Type	Default Value	Description
idProduct	REG_DWORD	0x0622	A bus-specific product identifier that associates a device with a Device Stage presentation, for example, DWORD:0622. For more information, see USB Function Client Driver Registry Settings (http://go.microsoft.com/fwlink/?LinkId=179334).
idVendor	REG_DWORD	0x045E	A vendor ID is the 4-digit vendor code that the USB committee assigns to the vendor. Remember to change this value to your vendor ID; 045E is reserved for use by Microsoft. For more information, see: • Device Identification Strings (http://go.microsoft.com/fwlink/?LinkId=146573) • Identifiers for USB Devices (http://go.microsoft.com/f

			wlink/?LinkID=179341) • USB Function Client Driver Registry Settings (http://go.microsoft.com/fwlink/?LinkID=179334) To obtain a vendor ID, see the USB Web site (http://go.microsoft.com/fwlink/?LinkID=119292). See also Standard USB Identifiers (http://go.microsoft.com/fwlink/?LinkID=179342).

If you are using the MTP IP transport, you must set the product description and the manufacturer URL (listed in Table 4) in the registry key:

HKEY_LOCAL_MACHINE\Software\Microsoft\MTP\Responder

Table 4 - IP Transport Registry Settings

Name	Type	Default Value	Description
ProductDescription	REG_SZ	Empty String/NULL	The user-friendly description of the device. For example, "Portable Navigation Device with 4.3 inch screen".
ManufacturerURL	REG_SZ	Empty String/NULL	The URL for your company's Web site.

Storage Configuration

For the user to be able to browse files on the device from their computer, you must define a single storage location for MTP objects (files and folders) on the device. You can specify the root of your device or one of its subdirectories as the MTP storage location.

To configure the MTP storage location on your device, configure the StorageRoot, StorageDescription, VolumeIdentifier, and AccessCapability settings (listed in Table 5) in the registry key:

HKEY_LOCAL_MACHINE\Software\Microsoft\MTP\Responder

Table 5 - Storage Registry Settings

Name	Type	Default Value	Description
StorageRoot	REG_SZ	Generated if not specified	The path to the MTP storage location. This path will appear as the root of the device's storage in Device Stage. If you do not specify the default value, MTP Responder will default to the My Documents folder. If the My Documents folder is not available, MTP Responder will use \MTPStorageRoot. For more information about storage representation on MTP devices, see the MTP 1.0 Specification (http://go.microsoft.com/fwlink/?LinkID=137101).
StorageDescription	REG_SZ	"MTP Storage"	A human-readable string, for example, "My Storage".
VolumeIdentifier	REG_SZ	"MTP Volume Identifier"	A unique identifier, such as a serial number. Only the first 128 characters are used to identify the device, and they must be unique for the device.
AccessCapability	REG_DWORD	ACCESS_CAPABILITY_READ_WRITE = 0	Identifies any write-protection characteristics that globally affect storage. You can set AccessCapability to one of the following values: ACCESS_CAPABILITY_READ_WRITE = 0 ACCESS_CAPABILITY_READ_ONLY_WITHOUT_OBJECT_DELETION = 1 ACCESS_CAPABILITY_READ_ONLY_WITH_OBJECT_DELETION = 2

Device Metadata Service

To transfer a custom presentation from the device to the computer when the device first connects to the computer, you must modify the Path, ContentID, and Flags registry settings (listed in Table 6) in the registry key:

HKEY_LOCAL_MACHINE\Software\Microsoft\MTP\Responder\Metadata\<Locale>

where **<Locale>** is the value from the **Locale** element in PackageInfo.xml in the device metadata package.

Table 6 - Metadata Service Registry Settings

Name	Type	Default Value	Description
Path	REG_SZ	No default	The absolute path on the system to the metadata package for the locale, for example Hard Disk1\WDS Metadata.
ContentID	REG_SZ	No default	A unique GUID string in the registry in CLSID format ("{xxxx-…}") that is assigned by the Windows Logo signing process, for example"{C23954F9-80A1-497d-AB9A-EFE0EFCEAA9C}".
Flags	REG_DWORD	No default	1 to designate that this package has the attribute <locale default="true"> in the PackageInfo.xml file in a device metadata package; otherwise, 0 (zero).

For a list of locale identifier strings, see Locale Identifier Constants and Strings (http://go.microsoft.com/fwlink/?LinkID=63026).

For more information about how Windows 7 uses the locales specified in device metadata packages, see *How to Localize Device Stage Experiences* in the Windows 7 Device Stage Reference Guide in the Windows Device Experience Development Kit (http://go.microsoft.com/fwlink/?LinkID=178109).

Step 4 Provide OAL Support for Device Stage

When a computer issues an MTP **GetDeviceInfo** request to your device, the MTP Responder gets information about the device from the SystemParametersInfo (http://go.microsoft.com/fwlink/?LinkID=179347) function.

To set that information, when you develop an OEM Adaptation Layer (OAL), you issue adaptation layer I/O control codes (IOCTLs) to inform the kernel of information about your device by calling the OEMIoControl (http://go.microsoft.com/fwlink/?LinkID=179348) function on device startup. You use the IOCTLS described below to set the manufacturer name and the device ID and add support for the battery level and free storage displays.

Manufacturer Name

To set the manufacturer name, use the following code, where *lpOutBuf* is a pointer to a buffer that contains the string representing your manufacturer name, and *lpBytesReturned* is the number of bytes that you wrote to lpOutBuf.

```
OEMIoControl(
        IOCTL_HAL_GET_DEVICE_INFO,
        SPI_GETPLATFORMMANUFACTURER,
        4,
        lpOutBuf,
        lpBytesReturned)
```

Device ID

To set the device ID, use the following code, where *lpOutBuf* is a pointer to a buffer that contains the string representing the device ID, and *lpBytesReturned* is the number of bytes that you wrote to *lpOutBuf*.

```
OEMIoControl(
        IOCTL_HAL_GET_DEVICE_INFO,
        SPI_GETUUID,
        4,
        lpOutBuf,
        lpBytesReturned)
```

For more information about the device ID, see Device ID (http://go.microsoft.com/fwlink/?LinkID=179344) and How do I get the "right" Device ID? (http://go.microsoft.com/fwlink/?LinkID=179345).

Battery Level

To add support for battery level, implement the **IOCTL_BATTERY_GETSYSTEMPOWERSTATUSEX2** control in your battery driver code. This control returns a SYSTEM_POWER_STATUS_EX2 structure that reports the battery level in the **BatteryLifePercent** field. Battery level is expressed as a percentage, with a value ranging from 0 to 100. The value that your battery driver code writes into **BatteryLifePercent** shows up as the battery level status for your device in the Devices and Printers folder on the computer. You must also enable **SYSGEN_BATTERY** to support battery level status in your device.

To retrieve the battery level from your device, Device Stage on the computer issues an MTP **GetDevicePropDesc** request to the MTP Responder on your device. The MTP Responder, in turn, calls the **GetSystemPowerStatusEx2** function to determine the current battery level to send back to Device Stage.

You must implement support for battery level status even if the device has no battery. For example if the device must be connected to an A/C outlet to operate, you still must implement support for battery level status. When plugged in to A/C power, the device will indicate a power level of 100 percent. If you do not include

GetSystemPowerStatusEx2 in your OS design, the device defaults to a power level of 100 percent.

For more information, see IOCTL_BATTERY_GETSYSTEMPOWERSTATUSEX2 (http://go.microsoft.com/fwlink/?LinkID=179349). For more about **GetSystemPowerStatusEx2**, see GetSystemPowerStatusEx2 (http://go.microsoft.com/fwlink/?LinkID=179350).

Free Storage Space

Add support for free storage space status if you are implementing a device file system for use with MTP Responder. To add support for free storage space status, ensure that **GetDiskFreeSpaceEx** properly returns the total capacity and the amount of free storage on your device.

To retrieve information about physical storage in your device, Device Stage issues an MTP **GetStorageInfo** request to the MTP Responder in your device. The MTP Responder, in turn, calls the **GetDiskFreeSpaceEx** function to determine the total capacity and the amount of free space to send back to Device Stage. This function returns information about the amount of space on the local file system of the device. **GetDiskFreeSpaceEx** returns the total amount of space and the total amount of free space; these quantities appear next to the image for your device in Device Stage.

For more information, see GetDiskFreeSpaceEx (http://go.microsoft.com/fwlink/?LinkID=179351).

Step 5 Create a Presentation for Windows 7

You can either use the *baseline* presentation for Device Stage, which is the default, or you can create a custom presentation.

Device Icon

When your device connects to a computer running Windows 7 for the first time, the computer attempts to load an icon file from your device and then store this icon in a Device Manager node on the computer. A device icon is recommended but not required for a baseline presentation for devices in the Portable Devices category. For custom presentations, the device icon is required in the metadata package, but it is not required on the device.

To add an icon to your device:

1. Create an icon file in the .ico file format. See the instructions for creating icons in the Windows 7 Device Stage Design Guide, which is in the Windows Device Experience Development Kit (http://go.microsoft.com/fwlink/?LinkID=178109).
2. Give your device icon the following name: MTPdeviceicon.ico.
3. Copy MTPdeviceicon.ico to the \Windows directory during binary image build. For more information about binary image build, see Binary Image Builder File http://go.microsoft.com/fwlink/?LinkID=179354).

Baseline Presentation

The baseline presentation is a good choice if you do not want to develop a customized metadata package and submit it to Microsoft to be signed. You can replace the baseline presentation later with a custom presentation.

The *baseline* presentation contains:

- Status information from device drivers, such as battery level, storage space, storage configuration.
- Information from the registry on your device, such as device friendly name, model and manufacturer name.
- A device icon from the \Windows directory on the device.

The following figure shows a baseline presentation in the Device Stage window.

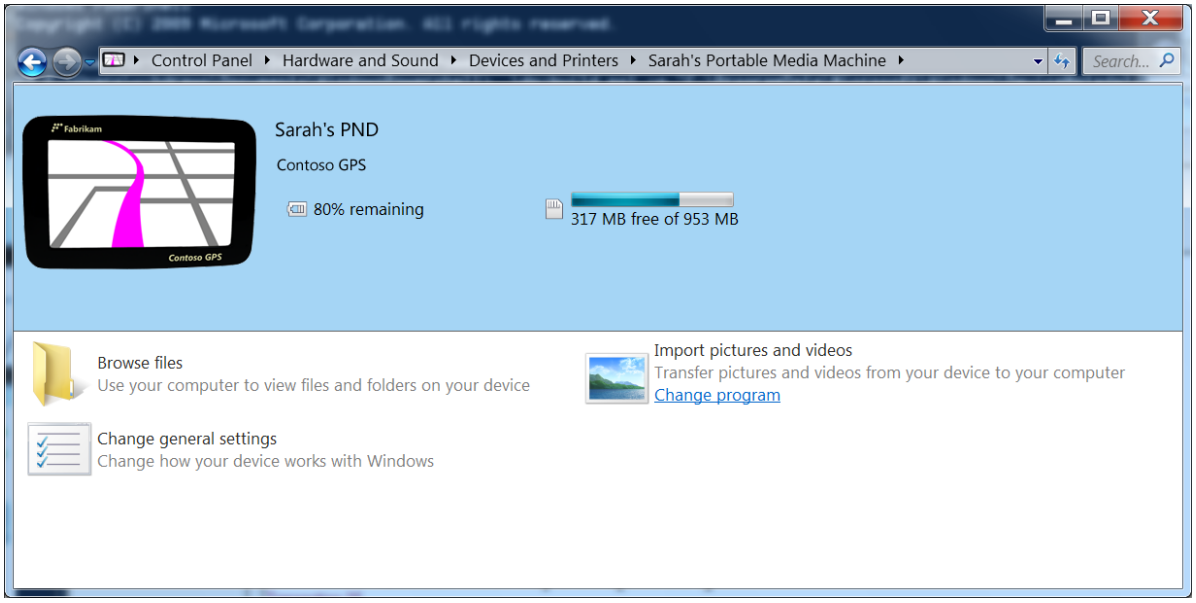

Figure 1 - Example Baseline Presentation

For more information about the Device Stage baseline presentation, see Windows 7 Device Stage Portable Device Class Development Guide in the Windows Device Experience Development Kit (http://go.microsoft.com/fwlink/?LinkID=178109).

Custom Presentation

Custom presentations can include your company's branding elements and custom tasks that are specific to your device. When you build a custom presentation, you must submit your device to Microsoft for Windows Logo certification. After your device is certified, you can submit your metadata packages to Microsoft to be signed. Device Stage only displays signed device metadata packages.

A custom presentation contains:

- Status information, which is the same as in the baseline presentation, unless you add custom status information.

- Information from XML files in a device metadata package that you create for your device.

The following figure shows a custom presentation in the Device Stage window that includes branding elements and custom tasks.

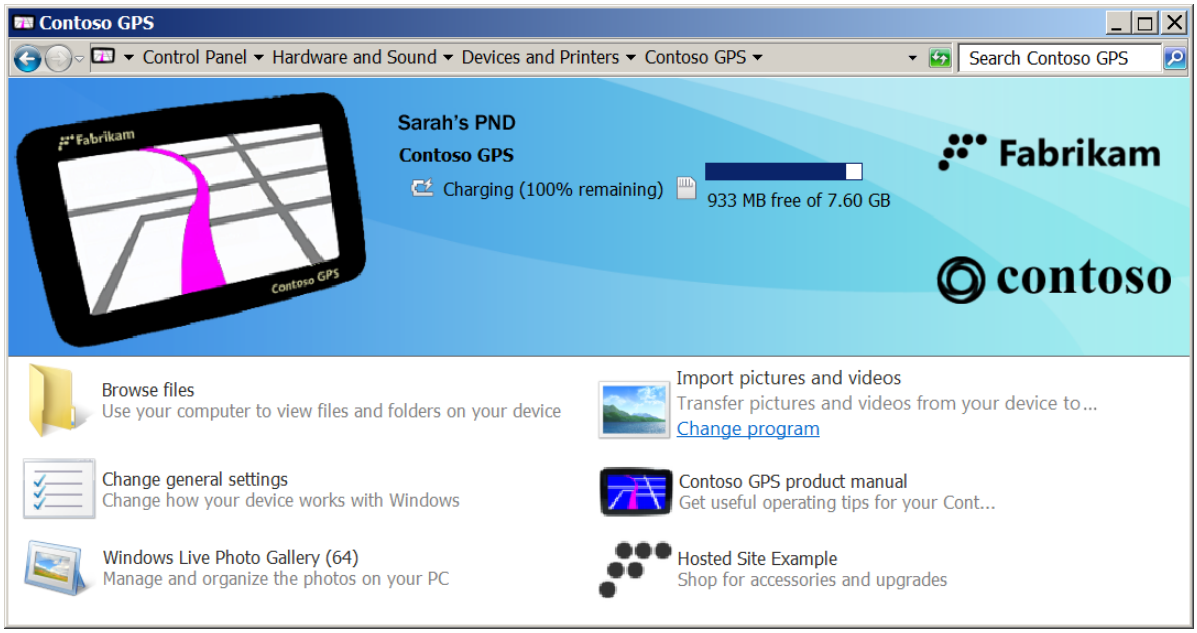

Figure 2 - Example Custom Presentation

For information about creating metadata packages that drive Custom Device Stage presentations, see the Windows Device Experience Development Kit (http://go.microsoft.com/fwlink/?LinkID=178109).

Step 6 Test Your Presentation

You can test your presentation on a computer running Windows 7.

Baseline

Use the following steps to verify that your baseline presentation functions as expected.

To test a baseline presentation

1. Connect your device to the computer using a USB cable.
2. If Device Stage does not launch, click **Start** and then click **Devices and Printers**.
3. In the **Devices and Printers** folder, identify the icon and text that represent your device.
4. Right-click the icon and click **Properties**.
5. On the **General** tab, verify that the displayed text is correct, and then click **Cancel**.
6. In the **Devices and Printers** folder, double-click the icon for your device.
7. In the Device Stage window, verify that the icon and text appear as you intended.

Custom

Device Stage only displays device metadata packages that have been signed by Microsoft. To test a custom metadata package, you must put Windows into a special test-signing mode.

To put Windows into test-signing mode

1. Click **Start** and type **cmd** (do not press **Enter**)
2. In the list of search results, right-click **cmd.exe,** and then click **Run as Administrator**.
3. In the command window, type **Bcdedit -set testsigning ON** and then press **Enter**. For more about this boot configuration setting, see TESTSIGNING Boot Configuration Option (http://go.microsoft.com/fwlink/?LinkId=179113)
4. Restart your computer.

The text "Test Mode Windows 7" displays on the desktop when Windows is in test-signing mode.

To test a custom presentation

1. Copy the device metadata package to C:\ProgramData\Microsoft\Windows\DeviceMetadataStore\<Locale>, where <Locale> is the value from the **Locale** element in PackageInfo.xml in the device metadata package.

   Note ProgramData is a hidden folder.
2. Follow steps 1-7 for the baseline presentation, above.
3. Verify that all the graphic elements, status displays, and tasks that you added to the device metadata package are displayed.
4. Verify that the status elements display correct values for your device.
5. Verify that all the tasks function as you expect.
6. Test all the links in your presentation.

Step 7 Complete Windows Logo Testing

You can have your device certified for the Windows logo if the device is in any of the following Device Stage Other Portable Devices categories.

- Portable navigation devices
- Consumer internet devices
- Digital picture frames
- E-book readers
- Portable gaming devices
- Set-top boxes

To qualify for Windows Logo certification, these devices must support battery level status and free storage status and provide functionality to browse files. In addition, there are requirements for devices in each category that you must meet to pass Windows Logo testing. For more information, see the Windows Logo site (http://go.microsoft.com/fwlink/?LinkId=8772).

Step 8 Submit Your Presentation to Windows Quality Online Services

See the Windows Quality Online Services site (http://go.microsoft.com/fwlink/?LinkID=179356) to submit your custom metadata package to Microsoft to be signed.

MTP Responder Design

MTP Responder for Windows Embedded CE 6.0 R3 is based on the MTP responder in the Windows 7 Portable Device Enabling Kit (DEK) for MTP, Version 7R2. This kit contains the source code for the reference implementations of an MTP initiator and an MTP responder. We derived MTP Responder from the reference MTP responder in this kit, adding functionality so that Windows Embedded CE 6.0 R3 devices support Device Stage operations.

MTP Responder Structure

The following figure shows the two major functional blocks of the MTP Responder: the *MTP Responder Stack* and *MTP Storage*.

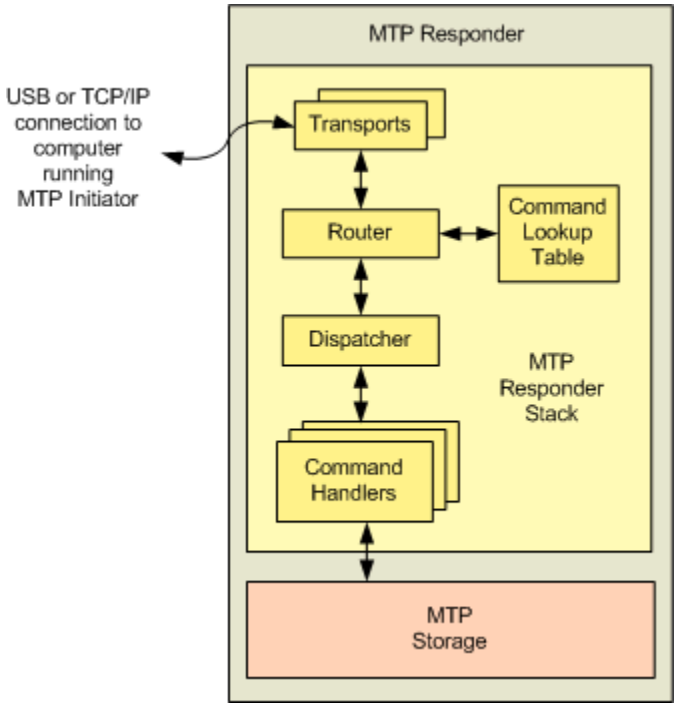

Figure 3 - MTP Responder

The MTP Responder Stack implements the core MTP operations, events, properties, and object formats of the Media Transfer Protocol (MTP). The MTP Responder Stack also implements an extension to the MTP specification that provides support for MTP Device Services. As shown on the left side of Figure 3, the MTP Responder Stack communicates with an MTP initiator through either a USB or TCP/IP connection. For MTP operations

that require access to files, metadata, or device properties, the MTP Responder Stack calls MTP Storage (shown at the bottom of Figure 3) to process these requests. When you add the MTP Responder (default) or MTP Responder (minimal) catalog items to your OS design, you include the MTP Responder Stack.

MTP Storage provides support for device files and folders so that users can browse and manage files on the device from a computer and copy files between the device and a computer. MTP Storage supports the MTP file formats, properties, and commands that are required for Device Stage. When you add the MTP Responder (default) catalog item to your OS design, you automatically include MTP Storage.

MTP Responder Stack

The MTP Responder Stack is made up of several subcomponents, as shown above in Figure 3.

- **Transport** Controls data transport across a USB or TCP/IP connection with the MTP initiator. MTP Responder for Windows Embedded CE 6.0 R3 includes two transports, one for USB and one for network connectivity.
- **Router** Routes MTP commands from each transport to the dispatcher, using the command lookup table to map each command to a route through the dispatcher and command handlers.
- **Command Lookup Table** Describes the route that an MTP command will take through the dispatcher and command handlers.
- **Dispatcher** Forwards MTP commands from the router to the appropriate command handler. In addition, if an MTP operation includes input data from the MTP initiator, the dispatcher routes the data for that operation to the command handler.
- **Command Handler** Implements the handler routines that process the various MTP operations supported by the MTP Responder. Each command handler processes a particular MTP operation.

MTP Responder includes source code for a number of general-purpose functions that are used extensively by the MTP Responder Stack. These general-purpose functions are referred collectively as the platform services layer (PSL).

For more information about the MTP Responder Stack and the platform services layer, see the Media Transfer Protocol Porting Kit (http://go.microsoft.com/fwlink/?LinkID=59975).

Source Code Modification

You can modify or extend the provided MTP Responder Stack implementation to add Device Stage features that are specific to your device. For example, you can add MTP Device Services extensions to support device tasks for features such as credentials provisioning or device firmware upgrade. You can also create your own event handlers to report transfer progress or device connection and disconnection. In some cases, you may find it necessary to customize the MTP Responder Stack to make performance or resource optimizations unique to your device or to add support for a new MTP transport protocol. To make custom modifications and extensions to the MTP Responder Stack for your device, see the MTP Responder common source code at the following location:

%_WINCEROOT%\public\datasync\oak\mtp\common\MTP

The source files at the location above are derived from the MTP responder reference implementation that is included in the Portable Device Enabling Kit; this implementation is not specific to Windows Embedded CE. These files include the core MTP Responder Stack functionality:

- MTP dispatcher
- MTP router
- MTP common command handler functionality
- MTP core command handler
- MTP command lookup table
- Common code to support MTP transports
- Support for USB, TCP/IP, and other transports
- Message queue library
- MTP format and properties library
- MTP event manager and event handlers
- MTP performance logger

In addition, MTP Responder Stack source code that is specific to Windows Embedded CE 6.0 R3 is available at the following location:

%_WINCEROOT%\public\datasync\oak\wince\src\MTP

These source files implement MTP extensions that integrate the MTP Responder Stack with Windows Embedded CE 6.0 R3:

- Functionality to report Device Stage information back to the MTP initiator, such as device product description, manufacturer name, serial number, model ID, device icon, and battery level.
- Platform Service Layer functionality that is unique to Windows Embedded CE, including atomic functions, debug utilities, memory management, thread management, timer utilities, sockets, and message queues.
- Implementation of the MTP status service; generation of reports about battery level and storage space by calling OAL functions (such as **GetDiskFreeSpaceEx**), returning the results to the MTP initiator.
- Implementation of the MTP device metadata service.
- MTP transport functionality for Windows Embedded CE USB and network devices.

The MTP Responder source code is written in ANSI C. Depending on your objectives and the extent of the modifications or extensions that you want to make to the MTP Responder, you can modify any or all of these source code locations.

Programming References

Several resources are available to help you understand the MTP Responder Stack implementation and to guide you in making modifications to the MTP Responder source code.

For more information about the MTP responder reference implementation, see the Media Transfer Protocol Porting Kit (http://go.microsoft.com/fwlink/?LinkID=59975).

For more about MTP operations, events, properties, and formats, see the MTP 1.0 Specification (http://go.microsoft.com/fwlink/?LinkID=137101).

For more about MTP Device Services extensions, see the MTP Device Services Extension Specification (http://go.microsoft.com/fwlink/?LinkID=178887).

For more information about the DEK, see the Windows 7 Portable Device Enabling Kit (http://go.microsoft.com/fwlink/?LinkID=178864).

MTP Storage

The MTP metadata database that comes with the device enabling kit (DEK) is implemented as an in-memory cache that holds the metadata for MTP objects while the device is powered on. This implementation is not suitable for an actual MTP responder (device) for two main reasons:

- It does not store object metadata persistently; when the responder (a device) is powered off, the metadata is deleted. MTP Storage uses the file system to persist all objects shown in Table 7 - MTP Storage Object Properties except for the Persistent Unique Object Identifier.
- It does not take into account memory usage on devices; storing all object metadata in memory can consume prohibitive amounts of device memory, possibly destabilizing the device.

Therefore, MTP Responder for Windows Embedded CE 6.0 R3 includes a custom MTP storage implementation that you can use as provided for production devices. With this functionality, devices can communicate with computers running Windows XP, Windows Vista, or Windows 7, although only Windows 7 has the Device Stage user interface.

MTP Storage Design

MTP Storage uses the same APIs to communicate with the MTP Responder Stack as the metadata database in the DEK. Unlike the metadata database in the DEK, storage in MTP Responder for Windows Embedded CE 6.0 R3 has three important requirements:

- It must have a small memory footprint.
- It must store only file and folder metadata.
- It must use the object metadata that is inherently stored in the MTP file system. This metadata persists between MTP sessions and between device power cycles.

Because of these requirements, MTP Storage allocates storage for MTP object metadata in two parts:

- A set of in-memory hash tables store object handles and file paths for MTP objects. Because they are created anew each time that an MTP session is created, object handles can be assigned to a non-persistent storage. Object handles are only viable while an MTP session is active and are the only MTP metadata property that is stored in the hash tables. This keeps the memory consumption of the hash tables to a minimum.
- All MTP object properties for files of the Undefined format except object handles are stored in the MTP file system and retrieved as needed. This design reduces processor cycles and memory consumption that would be required to load object metadata for each MTP session. Because this data is stored in the MTP file system, it remains on the device after the end of an MTP session and after the device is powered off. Unlike object handles, all the other properties of MTP

objects do not change between MTP sessions unless they are modified outside the MTP session. For example, a user modifies the content of an MTP store on a removable memory card, such as a Secure Digital (SD) card, by plugging the card directly into a computer and modifying its contents.

MTP Storage Implementation

MTP Storage provides functionality with which users browse and manage files on a device and transfer files between a device and a computer. The implementation of file formats, file properties, and MTP commands is limited to those that are required to achieve this functionality.

This functionality also qualifies devices for Windows Logo certification in the new Other Portable Devices category. All categories of devices besides Other Portable Devices require support for file formats, properties, and commands that are not included in the provided MTP storage implementation. You can expand the provided code and add support for additional MTP formats, properties, and commands to qualify your device in other categories. For more information about the requirements that devices must meet to qualify for a Windows logo, create an account on the Windows Quality Online Services site (http://go.microsoft.com/fwlink/?LinkID=179356) and see Prepared Requirements Reports (http://go.microsoft.com/fwlink/?LinkID=179359).

To implement MTP storage in the OS design on your device

1. Specify the storage location (see the section MTP Storage Location).
2. Specify the registry value for the HashBucketCount setting.
3. Specify the registry value for the MaxCustomHeapSize setting.

MTP Storage Object Formats

The MTP storage implementation in MTP Responder for Windows Embedded CE 6.0 R3 supports only two MTP object formats: Undefined (files that support a minimum set of properties) and Association (folders). The device may store many file formats (.mpg, .jpg, and so on), but all file formats on the device will behave as MTP objects with the Undefined format.

The following conditions apply:

- Any files that a user adds to a device become MTP objects with the Undefined format.
- File formats other than Undefined behave as Undefined format files after being transferred to the device.
- All file formats on the device have only the MTP object properties that are required for files of the Undefined format, plus additional properties supported in the DEK, which are listed below.

For example, an audio file on the initiator (a computer) typically has metadata associated with it (such as artist, rating, and album name) that is stored separately from the file and, in some cases, also within the file. The audio-specific metadata that is separate from the file will not be copied to the device because MTP storage does not support storing properties for files that are not of the Undefined format. MTP Storage cannot access metadata that is stored inside files.

Windows Media Player (WMP) cannot synchronize with or in any other way interact with the MTP files on the device because the files no longer have any of the properties that characterize media files. You can add support for media file formats by replacing the hash tables with your own database and interfacing it to the MTP Responder code.

MTP files of the Undefined format support only a specific subset of MTP object properties. For more information about MTP file formats and their properties, see the MTP 1.0 Specification.

The Association object format corresponds to a folder on a computer. There are two types of Association objects in MTP, Undefined and Generic. MTP storage supports only the Generic association type. For more information, see the MTP 1.0 Specification.

MTP Storage Object Properties

The MTP storage provided with MTP Responder for Windows Embedded CE 6.0 R3 includes only the file properties that the MTP protocol requires for all file formats and additional properties that are supported in the DEK. MTP properties for specialized file formats, such as media files, are not included. The following table lists the object properties that are supported by the MTP storage implementation.

Table 7 - MTP Storage Object Properties

Object property	MTP Datacode	Description
StorageID	0xdc01	The storage area on the device where the object is stored. In MTP storage, you designate only one StorageID on the device for the folders shared for MTP access. All MTP objects on the device have the same StorageID.
Object Format	0xdc02	An object's format determines the properties that it will support. The format is either 0x3000 (file) or 0x3001 (folder), stored implicitly in the MTP file system on the device.
ProtectionStatus	0xdc03	The status is either 0x0000 (No Protection) or 0x8001 (Read-only data); the mapping depends on whether the object is marked read-only in the MTP file system.
Object Size	0xdc04	The size of the data component of the object, in bytes. This property is ignored for folders.
Object File Name	0xdc07	The file name of the object, including the extension but not the path.

Parent Object	0xdc0b	The object handle of the parent folder, which must be an object of the format Association (0x3001). If the object is at the root, this value is 0xffffffff.
Persistent Unique Object Identifier	0xdc41	Although called the persistent unique object identifier (PUOID) in the MTP specification, the PUOID is not persistent between sessions in the MTP Storage implementation. The PUOID is a 128-bit version of the object handle. It is not used by the computer. It cannot be used to manage synchronization of data on the computer.
Name	0xdc44	The value returned for a file is the file name without path or extension. For a folder, the value returned is the folder name without the path. The value that is sent by Windows when creating an object is ignored.
Non-Consumable	0xdc4f	Indicates whether the object was transferred to the portable media player for storage only and is not available to be consumed (for example, played) by the portable device. The device returns 1 for files and 0 (zero) for folders. Windows does not send any values for this property, but such values would be ignored if sent.
Date Created	0xdc08	The date and time when the object was created; this value maps to the value in the MTP Responder file system on the device.
Date Modified	0xdc09	The date and time when the object was last altered; this value maps to the value in the file system.
Hidden	0xdc0d	Identifies whether an object is displayed to users or is hidden and only used by applications. This value maps to the value in the MTP Responder file system.
Association Type	0xdc05	Describes the kind of collection. A value of 1 indicates that the Association Type is a generic folder, which is the only Association Type that MTP storage supports. This value is not ever sent by Windows. MTP Storage ignores any values sent, returns 1 for

		folder, and returns 0 (zero) for a file.
AssociationDesc	0xdc06	Provides additional information about the Association Type. MTP Storage ignores any values sent, returns 0 (zero) for either a file or a folder. This value is never sent by Windows.

For more information about MTP properties, see the MTP 1.0 Specification.

MTP Storage Commands

MTP command functionality in MTP Responder for Windows Embedded CE 6.0 R3 is confined to commands that are necessary to support file browsing and management on the device and to transfer files between a device and a computer. The following table describes the commands used by MTP Storage.

Table 8 - MTP Storage Commands

MTP Command	Description
OpenSession	Opens a new MTP session for communication between the computer and device.
CloseSession	Closes an active session.
GetStorageIDs	Retrieves a list of storage IDs for the storage areas on the device.
GetStorageInfo	Retrieves a StorageInfo data set that describes a storage area on the device.
GetNumObjects	Retrieves a count of the number of MTP objects that are stored on the device.
GetObjectHandles	Retrieves an array of object handles that the computer can use to access the MTP objects that are stored on the device.
GetObjectInfo	Retrieves the ObjectInfo data set for the specified MTP object on the device.
GetObject	Retrieves the binary data component of the specified object on the device.
DeleteObject	Deletes an object or set of objects from the device.
SendObjectInfo	Sends an ObjectInfo data set to the device to prepare it to receive a new object.
SendObject	Sends the binary data component of an MTP object to the device.
FormatStore	Provides a quick way to purge the device of all content. Using this command is faster than individually deleting each of the objects in the store.

SetObjectProtection	Sets the write-protection status for the data object that is referred to in the first parameter to the value indicated in the second parameter.
MoveObject	Changes the location of an object on the device by changing the storage on which it is stored, changing the location in which it is located, or both.
CopyObject	Causes the device to create a copy of the target object and place that copy in a location that is indicated by the parameters of this operation.
UpdateObjectPropertyList	Sets the property list for a particular object that will be updated with a new binary object. Use this command to replace the binary data of an existing object.
DeleteObjectPropList	Removes the properties specified in the DeleteObjectPropList data set from the specified object or objects. If a property is not removable, it is returned to its default value.
GetObjectPropsSupported	Retrieves a list of object properties that the device supports for a class of MTP objects that share a particular object format.
GetObjectPropDesc	Retrieves an ObjectPropDesc data set that describes a particular property of a class of MTP objects that share a particular object format.
GetObjectPropValue	Retrieves the current value of an object property.
SetObjectPropValue	Sets the current value of an object property.
GetObjectPropList	Retrieves an ObjectPropList data set that contains a list of property values from the device.
SetObjectPropList	Sets a list of property values in the device.
SendObjectPropList	Sends an ObjectPropList data set to the device to prepare the device to receive a new object.

For more information about MTP commands, see the MTP 1.0 Specification.

File Transfer Messages

In MTP Responder for Windows Embedded CE 6.0 R3, the only file format supported on the device is Undefined. If a user attempts to copy files that are not of the Undefined format to the device, the user's computer may display a message indicating that the version of Windows and the version of Windows Media Player (WMP) that are installed on the user's computer have detected that the device does not support the format of the files for which transfer has been initiated.

The following table lists the responses of the operating system when a user initiates a copy operation that includes files whose format is other than Undefined.

Table 9 - MTP Storage File Transfer Messages

Windows Operating system/WMP version	Behavior
Windows XP/WMP 10	An error message appears. The file copy is not allowed. A user must upgrade to WMP version 11 for Windows XP to be able to copy files whose format is not Undefined.
Windows XP/WMP 11	No message appears; the files copy successfully.
Windows Vista/WMP 11	No message appears; the files copy successfully.
Windows 7/WMP 12	A message appears with each initiated file copy operation indicating that the user's computer has detected that the device does not support the format of the files for which transfer has been initiated. Users can choose to dismiss the message and copy the file. When multiple files are being copied, users can choose not to receive a message for each file in the copy operation.

MTP Storage Location

In MTP Storage, you designate only a single storage location on the device for MTP objects of the Undefined or Association format (see the section Storage Configuration). The computer will display only the contents of the designated MTP storage location. There is no mechanism that prevents all files of the Undefined format in the designated MTP storage location on the device from being displayed on the computer. For example, if you designate "\" as the MTP storage location on the device, then all files of the Undefined format in the "\Windows" directory will be displayed on the computer.

You can designate the MTP Storage location to be on removable storage, for example a Secure Digital (SD) card. However, there are no provisions in the code to detect that a removable storage device has been removed. As a result, discrepancies may exist between the contents of the designated MTP storage and the displayed contents on the computer. For example, a user removes an SD card from the device, inserts it into a slot on the computer and modifies its contents. If the user reinserts the card into the device while the same MTP session is active, then the computer will still display the original list of files, even though the files in the MTP storage location have changed.

Because you can designate only a single storage location for MTP objects, you cannot designate a subdirectory such as \DeviceStorage and also designate an SD card such as \Flash1 as the MTP Storage location. Support for multiple MTP Storage locations is not implemented. It is also not possible to associate two paths on the device to a single designated MTP Storage location.

Optimization

You can modify settings for the MTP storage to optimize it for your device, for example to manage memory usage and to optimize performance. To configure registry settings

in your OS design to support Device Stage, modify the datasync.reg file, which can be found at this location:

%_WINCEROOT%\public\datasync\oak\files\datasync.reg

Specifying the Size of the Hash Tables

You specify the size of the hash tables to optimize the tables' load and to control the number of collisions. We recommend that you specify the size of the hash table so that the maximum number of expected entries uses no more than 65% of the tables' capacity. If the load increases beyond this capacity, the number of collisions typically begins to rise significantly. In this case, hash collisions are added to a list so that the search performance moves from *O(1)* to *O(m)* where *m* is the size of the linear chain to be searched for the entry.

To change the hash table size, modify the HashBucketCount registry value (see Table 10) in the registry key:

HKEY_LOCAL_MACHINE\Software\Microsoft\MTP\ObjectStore

Table 10 – HashBucketCount

Name	Type	Default value	Description
HashBucketCount	REG_DWORD	1000	You specify the number of hash buckets to allocate to the hash tables with this registry value. You may want to increase or decrease the amount of memory allocated to the hash tables depending on the storage capacity of your device, the scenarios that your device supports, or other factors.

For more information about registry settings for MTP storage, see the section Storage Configuration.

Specifying the Size of the Private Heap

You can specify the size of the private heap that stores file and folder name strings to help prevent out-of-memory conditions on your device. When you specify the size of the private heap, consider how many files your users will store on the device and how often your users will rename, delete, or move files. For example, users may be more likely to rename photos than audio files.

To change the size of the private heap, modify the MaxCustomHeapSize registry value (see Table 11) in the registry key:

HKEY_LOCAL_MACHINE\Software\Microsoft\MTP\ObjectStore

Table 11 – MaxCustomHeapSize

Name	Type	Default value	Description
MaxCustomHeapSize	REG_DWORD	100000	Controls the size of the private strings heap. The private heap memory is reserved and then allocated dynamically from the reserved block. You specify the size of the reserved block in bytes.

Preventing Out-of-Memory Conditions

If the file system is extremely large and a user is traversing the depth and breadth of all directories on the device, then the user may reach the limits for in-memory storage of file path mapping to object handles.

When a user only browses files, memory will be consumed to keep track of the mapping of object handles to files and folders after a number of files and folders have been browsed.

When a user deletes, renames, or moves a large number of files, it alters the performance of the data structure and likely reduces the maximum number of files and folders that can be actively enumerated. Consider the following description of the memory implementation to guide your decisions for configuring the hash tables and private heap on your device.

- There is a fixed number of buckets in the hash tables, which you can specify.
- Each entry in a hash table must be allocated when inserted (these are small objects). MTP storage uses the global process heap for this task, and you cannot configure its size.
- There is a private heap that stores file name and folder name strings, whose size you can configure.
- When a user browses files and file enumeration occurs, MTP storage adds new entries to the private heap.
- When a user moves or renames a file, MTP storage adds a “delete” and a “create” entry to the private heap (causing minor fragmentation).
- When a user copies a file, MTP storage adds a single “create” entry to the private heap.
- When a user deletes a file, MTP storage may add many “delete” entries to the private heap (causing minor to major fragmentation).

If you find that your device encounters out-of-memory conditions when executing its scenarios, try increasing the values specified in the registry for HashBucketCount and MaxCustomHeapSize as indicated above. Increasing the value for HashBucketCount will improve performance because, for any given number of files, there will be fewer in each bucket that must be enumerated for file operations. Increasing the value for MaxCustomHeapSize will help prevent out-of-memory conditions. However, increasing the value for either HashBucketCount or MaxCustomHeapSize will consume more memory on the device.

MTP Storage Modifications or Replacement

If you want to modify the provided MTP storage implementation to include support for additional MTP object formats (files, folders, and so on), for additional MTP object properties, or for additional MTP commands, then you will need to make substantial modifications to the code. For example, to add support for file formats such as JPEG, MP3, or MPEG, or object properties such as Artist, Image Bit Depth, or Video Bit Rate you must replace the hash tables with another database and revise the code that interfaces the database to the MTP Responder stack.

If your devices and scenarios require support for file formats, file properties, or MTP commands that are not included in the MTP storage implementation, you can either:

- Replace the hash tables with your own database solution, for example a SQL CE database, and modify the provided code in MTP Responder for Windows Embedded CE 6.0 R3.
- Replace the hash tables with your own database solution, for example a SQL CE database, and author code that interfaces your database to the MTP Responder code in MTP Responder for Windows Embedded CE 6.0 R3.

For more about source code modifications, see the section Modifying the Provided Code. If you only need to support browsing, management, and transfers of Undefined format files for your device, then you can use MTP Storage in Windows Embedded CE 6.0 R3 as provided. You can add support for additional file formats, properties, or commands when using your own database if you want your device to qualify in a Windows Logo category besides Other Portable Devices.

Modifying the Provided Code

If you want to modify the MTP Storage code, the following libraries in the Windows Embedded CE 6.0 R3 platform code contain the bulk of the functionality that you may want to customize or extend:

%_WINCEROOT%\public\datasync\oak\mtp\wince\src\MTPObjectStoreShim

%_WINCEROOT%\public\datasync\oak\mtp\wince\src\MTPObjectStoreDB

Your code modifications may extend to files in other directories, depending on your objectives and the extent of the modification you want to make.

Replacing MTP Storage with Your Database

You can include the MTP Responder (minimal) and USB/IP transport support in your Device Stage OS design while omitting MTP Storage. With this option, you separately add your own database that you have interfaced to the MTP Responder code. See the section Step 2 Add Device Stage Catalog Items to Your OS Design.

Conclusion

Device Stage presents a graphical interface that makes it easy for users to find and use applications and services for their device when it is connected to a computer running Windows 7. MTP Responder for Windows Embedded CE 6.0 R3 provides the functionality for the device to communicate with Device Stage.

You can choose to implement a baseline presentation or a custom presentation for your device in Device Stage. The baseline presentation provides the most essential device

features and is simple to set up, whereas the custom presentation offers additional device features but it requires you to develop a device metadata package.

Because the MTP Responder code is available for modification, you can extend MTP Responder to support additional file formats and properties, add file synchronization, or replace the included MTP Storage component with your own storage implementation.

MTP Responder for Windows Embedded CE 6.0 R3 includes an MTP Storage component that you can use for production devices. You can configure MTP Storage settings to optimize performance and memory usage for your device, or you can modify the provided source code to integrate your own storage implementation to support MTP media file functionality that is not included in this implementation.

You can modify and extend the MTP Responder Stack to support additional MTP commands and properties. An MTP extension can define new MTP operations, properties, and object formats that are not part of the MTP specification.

Implementation of MTP Responder also prepares your device to meet the Windows Logo requirements for devices in the Other Portable Devices category.

Additional Resources

To learn more about Windows 7 Device Stage, see the following links.

- Windows Team Blog – The Device Experience in Windows 7 (http://go.microsoft.com/fwlink/?LinkID=179365)
- Windows Device Experience (http://go.microsoft.com/fwlink/?LinkId=132146)
- Windows Device Experience Development Kit (http://go.microsoft.com/fwlink/?LinkId=178109)
- Windows Logo Program (http://go.microsoft.com/fwlink/?LinkId=8772).

To learn more about Windows Embedded, see the following link.

- Windows Embedded Web site (http://go.microsoft.com/fwlink/?LinkId=25589)

Copyright