MTP Device Services for Windows

March 10, 2010

Abstract

This paper describes the services that Media Transfer Protocol (MTP) device services support on systems that run Windows® 7. MTP device services provide a framework for extending the functionality of the Windows Portable Devices (WPD) framework and provide new APIs that applications use to discover and access the extended functionality.

Device manufacturers can use this information to implement support for MTP device services. This paper assumes that the reader is familiar with WPD technology, the *USB Media Transfer Protocol Specification, Revision 1.0*, and the *MTP Device Services Extension Specification*.

This information applies to the Windows 7 operating system.

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

The current version of this paper is maintained on the Web at:   
 <http://www.microsoft.com/whdc/device/wpd/MTP-DevServ_Win7.mspx>

Disclaimer: This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

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

© 2010 Microsoft Corporation. All rights reserved.

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Change |  |  |  |
| March 10, 2010 | First publication | | | |

Contents

[1.0 Introduction 4](#_Toc255904374)

[2.0 Support for MTP Device Services in Windows 4](#_Toc255904375)

[2.1 Support for MTP Device Services in Windows 7 4](#_Toc255904376)

[2.2 Support for MTP Device Services in Windows Vista 4](#_Toc255904377)

[3.0 Using This Document 5](#_Toc255904378)

[3.1 Service Description Format 5](#_Toc255904379)

[3.1.1 General Information 5](#_Toc255904380)

[3.1.2 Service Properties 5](#_Toc255904381)

[3.1.3 Service Formats 5](#_Toc255904382)

[3.1.4 Properties for Each Service Format 6](#_Toc255904383)

[3.1.5 Stream Contents 7](#_Toc255904384)

[3.1.6 Methods 7](#_Toc255904385)

[3.1.7 Method Object Properties 7](#_Toc255904386)

[3.1.8 Method Parameters 7](#_Toc255904387)

[3.2 Dates and Times 8](#_Toc255904388)

[4.0 Services 8](#_Toc255904389)

[4.1 Contact Service 8](#_Toc255904390)

[4.1.1 Contact Service General Information 8](#_Toc255904391)

[4.1.2 Contact Service Properties 8](#_Toc255904392)

[4.1.3 Contact Service Formats 9](#_Toc255904393)

[4.2 Calendar Service 15](#_Toc255904394)

[4.2.1 Calendar Service General Information 16](#_Toc255904395)

[4.2.2 Calendar Service Formats 16](#_Toc255904396)

[4.2.3 Handling Recurring Appointments 22](#_Toc255904397)

[4.3 Notes Service 25](#_Toc255904398)

[4.3.1 Notes Service General Information 25](#_Toc255904399)

[4.3.2 Notes Service Properties 25](#_Toc255904400)

[4.3.3 Notes Service Formats 26](#_Toc255904401)

[4.4 Task Service 27](#_Toc255904402)

[4.4.1 Task Service General Information 27](#_Toc255904403)

[4.4.2 Task Service Properties 27](#_Toc255904404)

[4.4.3 Task Service Formats 27](#_Toc255904405)

[4.5 Status Service 31](#_Toc255904406)

[4.5.1 Status Service General Information 31](#_Toc255904407)

[4.5.2 Status Service Properties 31](#_Toc255904408)

[4.6 Hints Service 32](#_Toc255904409)

[4.6.1 Hints Service General Information 33](#_Toc255904410)

[4.7 Device Metadata Service 33](#_Toc255904411)

[4.7.1 Device Metadata Service General Information 33](#_Toc255904412)

[4.7.2 Device Metadata Service Formats 33](#_Toc255904413)

[4.8 Ringtone Service 34](#_Toc255904414)

[4.8.1 Ringtone Service General Information 34](#_Toc255904415)

[4.8.2 Ringtone Service Properties 35](#_Toc255904416)

[4.8.3 Ringtone Service Formats 35](#_Toc255904417)

[4.9 Enumeration Synchronization Service 36](#_Toc255904418)

[4.9.1 Enumeration Synchronization Service General Information 36](#_Toc255904419)

[4.9.2 Enumeration Synchronization Service Properties 36](#_Toc255904420)

[4.9.3 Enumeration Synchronization Service Formats 38](#_Toc255904421)

[4.9.4 Enumeration Synchronization Service Methods 39](#_Toc255904422)

[4.10 Anchor Synchronization Service 39](#_Toc255904423)

[4.10.1 Anchor Synchronization Service General Information 40](#_Toc255904424)

[4.10.2 Anchor Synchronization Service Properties 40](#_Toc255904425)

[4.10.3 Anchor Synchronization Service Formats 42](#_Toc255904426)

[4.10.4 Anchor Synchronization Service Methods 44](#_Toc255904427)

[5.0 Synchronizing PIM Data 45](#_Toc255904428)

[5.1 Selecting a Synchronization Service 45](#_Toc255904429)

[5.2 Enumeration Synchronization 45](#_Toc255904430)

[5.3 Anchor Synchronization 46](#_Toc255904431)

[5.4 More Information on PIM Synchronization 46](#_Toc255904432)

[6.0 Resources 46](#_Toc255904433)

# 1.0 Introduction

As described in the *USB Media Transfer Protocol Specification, Revision 1.0* (MTP specification), hardware vendors can specify extensions that define new MTP operations, properties, and object formats.

Although MTP 1.0 is an excellent protocol to facilitate the exchange of multimedia files and basic device command and control, it is not well-suited for newer, more advanced scenarios. The MTP device services architecture enables Windows® to locate and use various services and content types on a device. MTP device services define a new set of related operations, properties, and object formats in the form of a device service. By creating extensions to the Windows Portable Devices (WPD) API and the MTP protocol, Windows can locate, consume, and interact with useful content and services on the device. MTP device services for Windows are designed to support personal information manager (PIM) data, settings, and restricted-capabilities content types on a device. The resulting architecture also enables future platform extensibility.

To avoid conflicts in the limited 16-bit datacode space that is defined in the MTP specification, MTP device services use GUIDs to identify themselves to MTP initiators. Therefore, vendors can specify new device services independently of each other without the risk of defining service identifiers that conflict with each other. In addition, a device service can define its operations, properties, and formats independently of other device services.

This paper explains the services that MTP device services support. Although the device services framework offers a host of extensibility and new capabilities beyond Windows, this paper primarily addresses the services that are supported with inbox applications in Windows 7.

For more information on MTP, the MTP device services extension, and the WPD API, see “[Resources](#_Resources)” later in this paper.

# 2.0 Support for MTP Device Services in Windows

Some features are not available for all versions of Windows or might require additional software components.

## 2.1 Support for MTP Device Services in Windows 7

The WPD API and MTP support for device services are available on all versions of Windows 7 that support Windows Media® Player. To enable WPD, Windows Server® 2008 R2 users must install the Desktop Experience role and Windows 7 N customers must install the Media Feature Pack for Windows 7 N and Windows 7 KN.

## 2.2 Support for MTP Device Services in Windows Vista

In Windows Vista®, MTP device services are supported by installing the Platform Update for Windows Vista. If this update is installed, you can use the Hints service in Windows Vista as described in this paper. In Windows 7, Synchronization services, PIM services, Device Metadata service, and Status service all use the Device Stage™ user experience. You can use the WPD API to write a third-party application to provide similar functionality in Windows Vista.

# 3.0 Using This Document

This paper contains descriptions and datasets for MTP device services that are supported in Windows 7. Each service definition contains several required datasets to implement the service on a device. Use this paper in combination with the *MTP Device Services Extension Specification* on the WHDC Web site, which defines the MTP operations that are used to interact with device services.

Devices must provide mandatory components of the datasets and can provide optional components. Devices can provide components that this specification does not define, but Windows applications will not automatically support them.

## 3.1 Service Description Format

The following sections describe the information in the service descriptions in this paper. These descriptions also define the dataset in which to include each set of information.

### 3.1.1 General Information

This includes the service GUID, service name, and service type.

### 3.1.2 Service Properties

Service properties are defined as sets of service property PKey Namespaces and PKey IDs. You must include this information in the ServicePropDesc dataset.

The service descriptions include the following information for each service property.

|  |  |
| --- | --- |
| Namespace | The namespaces that apply to the property’s PKey. Assemble the full service PKey by combining the PKey Namespace and PKey ID. |
| ID | The ID that applies to the property’s PKey. Assemble the full service PKey by combining the PKey Namespace and PKey ID. |
| Name | The machine-readable name of the service property. |
| Type | The MTP datatype and form (if applicable) of the property. This is indicated in the form “MTP datatype – MTP form flag”. For example, an enumeration of unsigned 16-bit integers is represented as “UINT16 – ENUM (0x1)”. |
| R/W | Whether the service property is read-only (RO) or read/write (RW). |
| M/O | Whether the service property is mandatory (M) or optional (O). |
| Description | The description of the property. This is only for informational purposes and is not part of the service property dataset. |

### 3.1.3 Service Formats

This is a list of formats that Windows supports in the device service. You should include this information in the ServiceInfo dataset.

|  |  |
| --- | --- |
| GUID | The GUID of the format. |
| Name | The machine-readable name of the format. |
| R/W | Whether objects of this format should be read-only (RO) or readable/writeable (RW). Objects of read-only formats should carry a ProtectionStatus of 0x0001 (read-only). |
| M/O | Whether the format is mandatory (M) or optional (O). For the service to function correctly, mandatory formats must be implemented. |
| MIME Type | The MIME type of the format, if one exists. An MTP initiator can use this value to determine the functionality of formats that have no well-known format GUID. |
| Description | The description of the format. This is only for informational purposes and is not part of the ServiceInfo dataset. |

### 3.1.4 Properties for Each Service Format

After the service formats table, the service descriptions include information about each service format and include the object properties that apply to that particular format. You should implement the object properties in the ServiceCapabilities dataset. Some object properties are required, and others are optional. Where indicated, a property list applies to all formats in a service.

For Windows 7, association formats must take the 0x3001 FormatCode, the standard MTP format code.

Note that the following properties are generally required for all formats:

* StorageID  
  This must be implemented with the ObjectPropCode of 0xDC01 for all objects.
* ObjectFormat  
  This must be implemented with the ObjectPropCode of 0xDC02 for all objects.
* ParentID  
  Required.
* Name  
  If the PC does not send a name to the responder, the responder must generate an appropriate value for this property. The Name value can be used by Web scripting applications or Windows error messages.
* PUOID  
  This property is required unless otherwise specified.
* ObjectSize  
  Required.

The service descriptions include the following information about the properties of a service format.

|  |  |
| --- | --- |
| Namespace | The namespaces that applies to the object property’s PKey. Assemble the full object property PKey by combining the PKey Namespace and the PKey ID. |
| ID | The ID that applies to the object property’s PKey. Assemble the full object property PKey by combining the PKey Namespace and the PKey ID. |
| Name | The machine-readable name of the object property. |
| Type | The MTP datatype and form (if applicable) of the property. This is indicated in the form “MTP datatype – MTP form flag”. For example, an enumeration of unsigned 16-bit integers is represented as “UINT16 – ENUM (0x1)”. |
| R/W | Whether the object property is read-only (RO) or read/write (RW). DD indicates that the property may be read-only or read/write depending on the preference of the implementer. |
| M/O | Whether the object property is mandatory (M) or optional (O). |
| Description | The description of the property. This is only for informational purposes and is not part of the ServiceCapabilities dataset. |

### 3.1.5 Stream Contents

This section describes the binary data that is associated with an object, if applicable. For example, an abstract contact does not contain a binary data stream or file, but a VCARD contact contains a binary object with a VCARD-formatted data stream. Streams are accessed by using the GetObject, GetPartialObject, and SendObject operations.

### 3.1.6 Methods

This section describes the method formats that are used in the service. Each method format has information in both the ServiceInfo and ServiceCapabilities datasets. Method formats contain both object properties and method parameters, which are in the ServiceCapabilityList dataset.

The list of methods in a service description includes the following information for each method.

|  |  |
| --- | --- |
| GUID | The GUID of the method format. |
| Name | The name of the method format. |
| Associated Object | The object ID of an object that this method is acting against. If a method does not act against an object, this value is 0. |
| M/O | Whether the method must be implemented in the service. If optional (O), the method is not required. |
| Description | The description of the method and what it does. This is only for informational purposes and is not part of the ServiceCapabilities dataset. |

### 3.1.7 Method Object Properties

Similar to object format properties, each method format has multiple object properties that should be declared in the CapabilitiesList dataset by using the service object property extension form. A method has object properties and sometimes object parameters.

### 3.1.8 Method Parameters

The input and output parameters for a method are similar to object properties but have additional fields that indicate parameter ordering. Method parameters may contain streams (AUINT8) as inputs or outputs, which are described in the section that details the method’s operation. Parameter descriptions include the following information.

|  |  |
| --- | --- |
| PKey Namespace | The namespace that applies to the method parameter’s PKey. Assemble the full method parameter PKey by combining the PKey Namespace and PKey ID. |
| PKey ID | The ID that applies to the method parameter’s PKey. Assemble the full method parameter PKey by combining the PKey Namespace and PKey ID. |
| Name | The machine-readable name of the method. |
| Type | The MTP datatype and form (if applicable) of the property. This is indicated in the form “MTP datatype – MTP form flag”. For example, an enumeration of unsigned 16-bit integers is represented as “UINT16 – ENUM (0x1)”. |
| Param | The parameter number of the method parameter. |
| R/W | * RW if the parameter is read/write or in/out. * RO if the parameter is out-only. * WO if the parameter is in-only. * 0 if the parameter is the result parameter. |
| M/O | Whether the parameter is required for the method. |
| Description | The usage of the parameter. It is not part of the CapabilityList dataset. |

## 3.2 Dates and Times

Dates and times are used throughout this paper. All dates and times must comply with ISO8601 UTF16-LE. Unless specified, minute-resolution is required. Second-resolution may be indicated in some cases, but should not be used in Windows 7. Some properties may require only dates. All times should be in UTC time as the MTP specification requires.

# 4.0 Services

The following sections describe the MTP device services.

## 4.1 Contact Service

The Contact service contains the data for contacts and can optionally use a Synchronization service to facilitate synchronization with the PC. A device can have multiple contact stores, depending on how contacts are organized on the device. Examples of contact stores include SIM card contacts, work contacts, and personal contacts.

The Contact service consists of contact objects that have contact properties. The Windows synchronization functionality supports the properties that are specified in “[Contact Service Properties](#_4.1.2_Contact_Service)” later in this paper, but not all software applications support the full range of contact properties. Windows supports multiple contact formats, but devices must implement the “Contact” format. Mandatory properties are indicated in the table, whereas others may be implemented depending on the support desired.

#### Property Groups

When specifying property group codes, the following properties must be in GroupCode 0x1 (if property groups are implemented):

* PersonalPhone
* BusinessPhone
* MobilePhone
* All properties in the VersionProps dataset from the Synchronization service

### 4.1.1 Contact Service General Information

The general information for the Contact service includes the following:

* Service GUID: {DD04D5FC-9D6E-4F76-9DCF-ECA6339B7389} (SVCGUID\_Contacts)
* Service Name: Contacts
* Service Type: Default

### 4.1.2 Contact Service Properties

The service property namespace for the Contact service is listed in the following table.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {703d392c-532c-4607-9158-9cea742f3a16} | SYNCSVC\_SERVICE\_PROPERTIES | 1 |

The Contact service supports the following property.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/ O | Description |
| 1 | 4 | FilterType | UINT8 | RW | M | From the Synchronization service. This property is set by the synchronization host to track whether the content type–specific default synchronization filter is applied. If the filter is enabled, only contacts with phone numbers are synchronized. The device must keep this value, but is not required to understand it. |

### 4.1.3 Contact Service Formats

The Contact service supports the following service formats.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GUID | Name | R/W | M/O | MIME type | Description |
| {0xBB810000-AE6C-4804-98BA-C57B46965FE7} | Contact | RW | M | application/ octet-stream | Abstract contact file format |
| {0xBB820000-AE6C-4804-98BA-C57B46965FE7} | VCard2Contact | RW | O | text/x-vcard | VCARD file format (VCARD Version 2) |
| {0xBB830000-AE6C-4804-98BA-C57B46965FE7} | VCard3Contact | RW | O | text/x-vcard | VCARD file format (VCARD Version 3) |
| {0xBA060000-AE6C-4804-98BA-C57B46965FE7} | AbstractContactGroup | RW | O | application/ octet-stream | Generic format for contact group objects |

All Contact service formats support the following property namespaces.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {ef6b490d-5cd8-437a-affc-da8b60ee4a3c} | Generic Object Properties | 1 |
| {FBD4FDAB-987D-4777-B3F9-726185A9312B} | Contact Properties | 2 |
| {B28AE94B-05A4-4E8E-BE01-72CC7E099D8F} | Note Properties | 3 |

#### 4.1.3.1 Properties

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 3 | ParentID | UINT32 | RO | M | Parent ID. |
| 1 | 4 | Name | STRING | RW | M | Display name for this object. Generated by the device. Recom­mended to be Firstname + Lastname (for example, “John Doe”). |
| 1 | 5 | PUOID | UINT128 | RO | M | Persistent ObjectUnique ID. This must be a GUID. |
| 1 | 6 | ObjectFormat | UINT16 | RO | M | MTP format code that this object represents. |
| 1 | 11 | ObjectSize | UINT64 | RO | O | Size of this object in bytes. Typically, contact objects are abstract (contain only properties), so the size is 0. |
| 1 | 23 | StorageID | UINT32 | RO | M | Storage ID for this object. |
| 1 | 40 | DateModified | STRING - DateTime | RW | O | Date at which this object was last modified. |
| 1 | 28 | Hidden | UINT16 - ENUM | RO | O | Contact objects should not be hidden. Objects that are marked as hidden are enumer­ated for syn­chroniza­tion just the same as objects that are not marked hidden. |
| 2 | 3 | GivenName | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | M | First name of the contact. |
| 2 | 4 | MiddleNames | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Middle name of the contact. |
| 2 | 5 | FamilyName | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | M | Last name of the contact. |
| 2 | 6 | Title | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Prefix of the name of the contact. |
| 2 | 7 | Suffix | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Suffix of the name of the contact. |
| 2 | 8 | PhoneticGivenName | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Phonetic guide for pronouncing the contact’s first name. |
| 2 | 9 | PhoneticFamilyName | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Phonetic guide for pronouncing the contact’s last name. |
| 2 | 10 | PersonalAddressFull | String OR AUINT16 – LONGSTRING FORM (0xFF) | RW | O | Full personal postal address (for example, “123 Oak St, Redmond, WA 98052”). |
| 2 | 11 | PersonalAddressStreet | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | First line of a postal address of the contact (for example, “555 Dial Drive”). |
| 2 | 12 | PersonalAddressLine2 | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Second line of a postal address of the contact. |
| 2 | 13 | PersonalAddressCity | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | City of a postal address of the contact (for example, “Phoneland”). |
| 2 | 14 | PersonalAddressRegion | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Region of a postal address of the contact. |
| 2 | 15 | PersonalAddressPostalCode | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Postal code of an address of the contact. |
| 2 | 16 | PersonalAddressCountry | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Country of an address of the contact. |
| 2 | 17 | BusinessAddressFull | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Full business postal address (for example, “123 Oak St, Redmond, WA 98052”). |
| 2 | 18 | BusinessAddressStreet | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | First line of a postal address of the contact (for example, “555 Dial Drive”). |
| 2 | 19 | BusinessAddressLine2 | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Second line of a postal address of the contact. |
| 2 | 20 | BusinessAddressCity | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | City of a postal address of the contact (for example, “Phoneland”). |
| 2 | 21 | BusinessAddressRegion | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Region of a postal address of the contact. |
| 2 | 22 | BusinessAddressPostalCode | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Postal code of an address of the contact. |
| 2 | 23 | BusinessAddressCountry | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Country of an address of the contact. |
| 2 | 24 | OtherAddressFull | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Full additional postal address (for example, “123 Oak St, Redmond, WA 98052”). |
| 2 | 25 | OtherAddressStreet | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | First line of a postal address of the contact (for example, “555 Dial Drive”). |
| 2 | 26 | OtherAddressLine2 | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Second line of a postal address of the contact. |
| 2 | 27 | OtherAddressCity | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | City of a postal address of the contact (for example, “Phoneland”). |
| 2 | 28 | OtherAddressRegion | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Region of a postal address of the contact. |
| 2 | 29 | OtherAddressPostalCode | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Postal code of an address of the contact. |
| 2 | 30 | OtherPostalAddressCountry | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Country of an address of the contact. |
| 2 | 31 | Email | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | E-mail address for the contact (for example, “someone@example.com”). |
| 2 | 32 | PersonalEmail | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Personal e-mail address for the contact (for example, “someone@example.com”). |
| 2 | 33 | PersonalEmail2 | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Secondary personal e‑mail address for the contact (for example, “someone@example.com”). |
| 2 | 34 | BusinessEmail | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Business e-mail address for the contact (for example, “someone@example.com”). |
| 2 | 35 | BusinessEmail2 | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Secondary business e‑mail address for the contact (for example, “someone@example.com”). |
| 2 | 36 | OtherEmail | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Other e-mail address for the contact (for example, ”someone@example.com”). |
| 2 | 37 | Phone | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Phone number for the contact. |
| 2 | 38 | PersonalPhone | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | M | Personal phone number for the contact. |
| 2 | 39 | PersonalPhone2 | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Secondary personal phone number for the contact. |
| 2 | 40 | BusinessPhone | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | M | Business phone number for the contact. |
| 2 | 41 | BusinessPhone2 | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Secondary business phone number for the contact. |
| 2 | 42 | MobilePhone | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | M | Mobile phone number for the contact. |
| 2 | 43 | MobilePhone2 | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Secondary mobile phone number for the contact. |
| 2 | 44 | PersonalFax | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Personal fax number for the contact. |
| 2 | 45 | BusinessFax | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Business fax number for the contact. |
| 2 | 46 | Pager | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Pager number for the contact. |
| 2 | 47 | OtherPhone | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Other phone number for the contact. |
| 2 | 48 | WebAddress | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Web address for the contact. |
| 2 | 49 | PersonalWebAddress | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Personal Web address for the contact (for example, “http://www.example.com”). |
| 2 | 50 | BusinessWebAddress | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Business Web address for the contact (for example, http://www.example.com). |
| 2 | 51 | IMAddress | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Instant messenger address for the contact. |
| 2 | 52 | IMAddress2 | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Additional instant messenger address for the contact. |
| 2 | 53 | IMAddress3 | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Additional instant messenger address for the contact. |
| 2 | 54 | Organization | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Organization or company name for the contact. |
| 2 | 55 | PhoneticOrganization | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Phonetic guide for pronouncing the contact’s organization or company name. |
| 2 | 56 | Role | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Role for the contact (for example, “Software Engineer”). |
| 2 | 58 | Fax | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Primary fax number for the contact. |
| 2 | 59 | Spouse | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Full name of the spouse of the contact. |
| 2 | 60 | Children | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Array of type UTF‑16LE, where each element is the full name of a child of the contact. |
| 2 | 61 | Assistant | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Full name of the assistant of the contact. |
| 2 | 65 | Birthdate | STRING | RW | O | Birth date for the contact. No time should be included, only the date. Date must comply with ISO8601 UTF16-LE. |
| 2 | 66 | AnniversaryDate | STRING | RW | O | Anniversary date for the contact. No time should be included, only the date. Date must comply with ISO8601 UTF16-LE. |
| 2 | 63 | Ringtone | UINT32 – ObjectID FORM (0x83) | RW | O | Object ID of a ringtone in the ringtone service to be played when a phone call (or similar action) is received. |
| 3 | 3 | Body | AUINT16 - LONGSTRING FORM (0xFF) | RW | O | A note or memo about the contact. |

#### 4.1.3.2 Stream Contents

For abstract contact items, the binary stream contains no data. If VCard formats are used, the stream contains the VCard data in the appropriate format.

When VCard formats are used, the accompanying VCard string is not guaranteed to be updated during synchronization operations. It is the responsibility of the MTP responder to guarantee that VCard data and object properties remain synchronized.

## 4.2 Calendar Service

The Calendar service contains the data for a calendar and uses a Synchronization service to facilitate synchronization with the PC. Windows supports only one calendar for each device.

#### Property Groups

When specifying property group codes, the following properties must be in GroupCode 0x1:

* BeginDateTime
* EndDateTime
* All properties in the VersionProps dataset from the synchronization service

### 4.2.1 Calendar Service General Information

The general information for the Calendar service includes the following:

* Service GUID: {E4DFDBD3-7F04-45E9-9FA1-5CA0EAEB0AE3} (SVCGUID\_Calendar)
* Service Name: Calendar
* Service Type: Default

### 4.2.2 Calendar Service Formats

The calendar service supports the following service formats.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GUID | Name | R/W | M/O | MIME type | Description |
| {bf70e114-3901-4449-bee7-d9ea1493c309} | AbstractActivity | RW | M | application/ octet-stream | An abstract appointment. |
| {E87A7008-32D1-42C5-8488-4C235866AF32} | AbstractActivityOccurrence | RW | M | application/ octet-stream | An abstract appoin­t­ment occurrence. |
| {23F7A5A5-F7D3-4585-A1FF-76E2D45C9121} | vCalendar1 | RW | O | text/calendar | vCalendar file format (vCalendar Version 1). |
| {CC4538CB-7890-41B7-A3F1-B6E60BDD2A61} | iCalendar | RW | O | text/calendar | iCalendar file format (vCalendar Version 2). |

The following sections provide more information about the service formats.

#### 4.2.2.1 AbstractActivity Format

The AbstractActivity format supports the following property namespaces.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {ef6b490d-5cd8-437a-affc-da8b60ee4a3c} | Generic Object Properties | 1 |
| {F99EFD03-431D-40D8-A1C9-4E220D9C88D3} | Calendar Service Object Properties | 2 |
| {B28AE94B-05A4-4E8E-BE01-72CC7E099D8F} | Information Object Properties | 3 |

##### 4.2.2.1.1 Properties

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 3 | ParentID | UINT32 | RO | M | Parent ID. |
| 1 | 4 | Name | STRING | DD | M | Display name for this object, which is generated by the device. We recom­mend the Subject property value. This property is shown to the user in error messaging. |
| 1 | 5 | PUOID | UINT128 | RO | M | Persistent object unique ID. This must be a GUID. |
| 1 | 6 | ObjectFormat | UINT16 | RO | M | MTP format code that this object represents. |
| 1 | 11 | ObjectSize | UINT64 | RO | O | Size of this object in bytes. For abstract calendar objects, this is typically 0. For vCalendar/iCalendar objects, this contains the size of the vCalendar/iCalendar data stream. |
| 1 | 23 | StorageID | UINT16 | RO | M | Storage ID for this object. |
| 1 | 28 | Hidden | UINT16 - ENUM | RO | O | Calendar objects can be marked as hidden. However, the synchroniza­tion behavior during FullEnumeration synchronization does not ignore hidden objects. |
| 1 | 40 | DateModified | STRING - DateTime | RW | O | Date on which this object was last modified. |
| 3 | 2 | Subject | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | M | Subject of the calendar item. |
| 3 | 3 | Body | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Message or note that is associated with this calendar item. |
| 2 | 3 | Location | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | M | Location of the appointment (for example, “Building 5, Room 7”). |
| 3 | 10 | Category | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of categories that apply to the appointment. |
| 3 | 18 | Sender | STRING OR AUINT16 – LONGSTRING FORM (0xFF) | RW | O | SMTP address of meeting organizer. |
| 3 | 20 | To | AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of required attendees (SMTP addresses). |
| 3 | 21 | CC | AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of optional attendees (SMTP addresses). |
| 3 | 22 | BCC | AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of resources for this meeting (SMTP addresses). |
| 2 | 10 | Accepted | AUINT16 – LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of attendees who have accepted the appointment (SMTP addresses). |
| 2 | 12 | Tentative | AUINT16 – LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of attendees who have tentatively accepted the appointment (SMTP addresses). |
| 2 | 19 | BeginDateTime | STRING - DateTime | RW | M | Starting time for the appointment, in UTC time. For all-day events, only the date portion of the MTP Date field is valid. (Specifying the time as 0:00 is considered a start time, not a start date.) Date must comply with ISO8601 UTF16-LE. |
| 2 | 20 | EndDateTime | STRING - DateTime | RW | M | End time for the appointment, in UTC time. For all-day events, only the date portion of the MTP Date field is valid. (Specifying the time as 0:00 is considered an end time, not an end date.) Date must comply with ISO8601 UTF16-LE. |
| 2 | 18 | PatternDuration | UInt32 | RW | M | Duration of the appointment in minutes |
| 2 | 15 | ReminderOffset | UInt32 | RW | O | Number of minutes before the start time of the appoint­ment that a reminder is to be fired. |
| 3 | 27 | PatternType | UInt32- ENUM | RW | M | Enumeration that indicates the type of the recurring appointment. Acceptable values are Daily = 0x01, Weekly = 0x02, Monthly = 0x03, Yearly = 0x04 |
| 3 | 28 | PatternValidStartDate | String – ISO 8601 Date form | RW | M | Date at which the recurrence pattern becomes valid. All occurrences of the recurring appointment must occur on or after this date. |
| 3 | 29 | PatternValidEndDate | String – ISO 8601 Date form | RW | M | Date at which the recurrence pattern is no longer valid. All occurrences of the recurring appointment must occur before or on this date. Only one of PatternValidEndDate and PatternValidCount may be set. |
| 3 | 31 | PatternPeriod | UInt32 | RW | M | The period for the repetition of the recurrence pattern. The units of this value are deter­mined by the recurrence type. If recurrence type is Daily, this is a count of days; if recur­rence type is Weekly, this is a count of weeks; and so on. |
| 3 | 32 | PatternDayOfWeek | UInt16 | RW | M | For weekly, monthly and yearly recurrence patterns, the day of the week on which the pattern is valid. The field is a bitmask of the following values:   * Sunday = 0x0001 * Monday = 0x0003 * Tuesday = 0x0004 * Wednesday = 0x0008 * Thursday = 0x0010 * Friday = 0x0020 * Saturday = 0x0040 |
| 3 | 33 | PatternDayOfMonth | UInt8 | RW | M | For monthly and yearly recurrence patterns, the day of the month on which the pattern is valid. The last day of the month is always indicated by 31, even on months that do not have 31 days. |
| 3 | 34 | PatternMonthOfYear | UInt8 | RW | M | For yearly recurrence patterns, the month of the year on which the pattern is valid. |
| 3 | 35 | PatternInstance | UInt8 | RW | M | For monthly and yearly recurrence patterns, the instance of the pattern that matches:   * 0x01 = first instance * 0x02 = second instance * 0x05 = last instance |
| 3 | 36 | PatternDeleteDates | AUINT16 – LONGSTRING FORM | RW | M | A list of dates on which the recurrence pattern does not apply. This list contains only dates that have been deleted, not dates that have been moved. |
| 2 | 14 | TimeZone | STRING | RW | M | The time zone in which the appointment was created. Values for this field must come from the public-domain tz database ([http://www.twinsun.com /tz/tz-link.htm](http://www.twinsun.com/tz/tz-link.htm)). |
| 2 | 16 | BusyStatus | UInt16 | RW | O | How the appointment is to appear on the calendar:   * 0x0000 = Free * 0x0001 = Busy * 0x0002 = Out of Office * 0x0003 = Tentative |
| 2 | 17 | PatternStartTime | String – ISO 8601 Time form | RW | M | The time of day in local time that the recurring appoint­ment is to occur. |

##### 4.2.2.1.2 Stream Contents

If the abstract calendar format is used, the stream is empty. If either the vCalendar or iCalendar format is used, the stream contains the appropriately formatted data.

Note that synchronization operations operate only on abstract objects. If vCalendar or iCalendar formats are used, it is the responsibility of the MTP responder to keep the properties and the vCalendar/iCalendar streams synchronized.

#### 4.2.2.2 AbstractActivityOccurrence Format

The AbstractActivityOccurence format supports the following property namespaces.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {ef6b490d-5cd8-437a-affc-da8b60ee4a3c} | Generic Object Properties | 1 |
| {F99EFD03-431D-40D8-A1C9-4E220D9C88D3} | Calendar Service Object Properties | 2 |
| {B28AE94B-05A4-4E8E-BE01-72CC7E099D8F} | Information Object Properties | 3 |

##### 4.2.2.2.1 Properties

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 3 | ParentID | UINT32 | RO | M | Parent ID for this object. Because occurrence objects are generically stored at the root of the service, this is typically 0. |
| 1 | 4 | Name | STRING | RW | M | Display name for this object, which the device generates. We recom­mend the Subject property value. This property is shown to the user in error messaging. |
| 1 | 5 | PUOID | UINT128 | RO | O | Persistent object unique ID, which must be a GUID. This is optional because occur­rences are refer­enced by activities. |
| 1 | 6 | ObjectFormat | UINT16 | RO | M | MTP format code that this object represents. |
| 1 | 11 | ObjectSize | UINT64 | RO | O | Size of this object in bytes. Because this format has no associated stream data, it should be 0. |
| 1 | 23 | StorageID | UINT32 | RO | M | Storage ID for this object. |
| 1 | 28 | Hidden | UINT16 - ENUM | RO | O | Occurrence objects, which should typically be marked as hidden because their enumeration should depend on the parent appoin­­t­ment that refer­ences them. |
| 1 | 40 | DateModified | String – DateTime Form | RW | O | Date on which this object was last modified. |
| 3 | 2 | Subject | STRING OR AUINT16 – LONGSTRING FORM (0xFF) | RW | M | Subject of the calendar item. |
| 3 | 3 | Body | AUINT16 – LONGSTRING FORM (0xFF) | RW | O | Message or note that is associated with this calendar item. |
| 2 | 3 | Location | STRING OR AUINT16 – LONGSTRING FORM (0xFF) | RW | M | Location of the appointment (for example, “Building 5, Room 7”). |
| 3 | 10 | Category | STRING OR AUINT16 – LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of categories that apply to the appointment. |
| 3 | 18 | Sender | STRING OR AUINT16 – LONGSTRING FORM (0xFF) | RW | O | SMTP address of meeting organizer. |
| 3 | 20 | To | AUINT16 – LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of required attendees (SMTP addresses). |
| 3 | 21 | CC | AUINT16 – LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of optional attendees (SMTP addresses). |
| 3 | 22 | BCC | AUINT16 – LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of resources for this meeting (SMTP addresses). |
| 2 | 10 | Accepted | AUINT16 – LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of attendees who have accepted the appointment (SMTP addresses). |
| 2 | 12 | Tentative | AUINT16 – LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of attendees who have tentatively accepted the appointment (SMTP addresses). |
| 2 | 19 | BeginDateTime | STRING – DateTime | RW | M | Starting time for the appointment in UTC time. For all-day events, only the date portion of the MTP Date field is valid. (Specifying the time as 0:00 is considered a start time, not a start date.) Date must comply with ISO8601 UTF16-LE. |
| 2 | 20 | EndDateTime | STRING - DateTime | RW | M | End time for the appointment, in UTC time. For all-day events, only the date portion of the MTP Date field is valid. (Specifying the time as 0:00 is considered an end time, not an end date.) Date must comply with ISO8601 UTF16-LE. |
| 2 | 18 | PatternDuration | Uint32 | RW | M | Duration of the appointment in minutes. |
| 2 | 15 | ReminderOffset | Uint32 | RW | O | Number of minutes before the start time of the appoint­ment that a reminder is to be fired. |
| 2 | 16 | BusyStatus | Uint16 | RW | O | How the appointment is to appear on the calendar:   * 0x0000 = Free * 0x0001 = Busy * 0x0002 = Out of Office * 0x0003 = Tentative |
| 3 | 26 | PatternOriginalDateTime | String – DateTime Form | RW | M | The original time that this appointment occurred in the recurrence set. The time should be encoded as a non-biased time in the time zone of the appoint­ment. For example, if the occurrence was to originally happen on 06/24/2009 at 10:00 AM local time, the DateTime value is 06242009T100000. |
| 1 | 47 | ReferenceParentID | UInt32 ObjectID | W | M | Object ID of the parent AbstractActivity appoint­ment with which this object is associated. This property is a write-only property and is intended to be used only when creating new occur­rence objects as a hint to the MTP responder of the parent object with which it is associated. |

##### 4.2.2.2.2 Stream Contents

The contents of the occurrence stream are empty.

### 4.2.3 Handling Recurring Appointments

The Calendar service requires support for recurring appointments. These appointments are represented by a single appointment object but might have referenced occurrence objects for handling changes to items in the recurrence set.

Managing recurrence patterns can be complicated because the meaning of the pattern is tied to the original time zone of the appointment. All dates and times in the recurrence pattern properties are considered local to the time zone that is specified in the TimeZone property. When you perform operations on these dates, you must convert to UTC or to the current local time at the final stage of processing.

The rest of this section describes recurrence patterns and handling moved occurrences.

#### 4.2.3.1 Recurrence Patterns

MTP device services support daily, weekly, monthly, and yearly recurrence patterns. Some properties are generically applied to all recurrence types, and some properties apply only to specific types of recurrence patterns. The set of supported patterns is much more limited than the iCalendar specification supports, but it is sufficient to handle most common recurrence patterns.

##### 4.2.3.1.1 PatternType and PatternPeriod Properties

The PatternType property applies to all recurring appointments and defines the type of recurrence that is being represented. The presence of the recurrence type property, regardless of its value, indicates that the appointment is a recurring appointment. The PatternType can be daily, weekly, monthly, or yearly. For a list of the recurrence patterns that MTP device services support, see “[Recurrence Patterns Reference](#_Recurrence_Patterns_Reference)” later in this paper.

The period of the recurring appointment is specified in the PatternPeriod property. This property is always an integer value in the units of the pattern type. For example, if PatternType is daily, the unit of the value of PatternPeriod is days and a value of 4 indicates a period of 4 days. If PatternType is monthly, the unit of PatternPeriod is months and a value of 4 indicates a period of 4 months.

##### 4.2.3.1.2 PatternValid*Nnn*Date and PatternValidCount Properties

The PatternValidStartDate property applies to all recurring appointments. It defines the first date on which the recurrence pattern might be valid. Only dates can be specified. It is not legal to specify a time. PatternValidStartDate does not necessarily indicate the date of the first occurrence. It indicates only the earliest date on which the occurrence might appear.

The PatternValidEndDate or PatternValidCount properties manage the number of occurrences that the pattern specifies. These properties are mutually exclusive: only one can appear on an appointment at a time. For better performance, developers should use the PatternValidEndDate property to specify the number of occurrences. Like PatternValidStartDate, PatternValidEndDate might not actually represent a date in the recurrence pattern. All occurrences in the pattern occur on or before PatternValidEndDate. PatternValidCount specifies the actual number of occurrences that are to appear on the calendar. Note that this is the total number of occurrences, not just the number of remaining occurrences.

Adjustments to the recurrence pattern do not require adjustment of PatternValid*Nnn*Dates or PatternValidCount.

##### 4.2.3.1.3 PatternDeleteDates Property

It is possible that not all occurrences of a recurring appointment are valid. If the user deletes an occurrence, the date of the occurrence is added to the semicolon-delimited list of dates that are stored in the PatternDeleteDates property. These dates must be specified in the time zone of the appointment and not in the time zone of the device or in UTC time. For example, if the user deletes an occurrence of a recurring appointment that is scheduled for 8:00 PM on 04/18/2010 in the time zone that is specified in the TimeZone property but at 4:00 AM on 04/19/2010 in UTC time, the date that is specified in the PatternDeleteDates property is 04182010.

If the user modifies an occurrence of a recurring appointment, the original date of the occurrence is not included in the PatternDeleteDates property.

##### 4.2.3.1.4 Recurrence Pattern Reference

MTP device services support the recurrence patterns in the following tables. Each table includes a description of each pattern and a list of the properties that apply to each pattern.

Table 1. Daily Recurrence Pattern

|  |  |
| --- | --- |
| Pattern | Properties |
| “Repeat every N days” | PatternType = Daily  PatternPeriod = N |

Table 2. Weekly Recurrence Patterns

|  |  |
| --- | --- |
| Patterns | Properties |
| “Repeat every N weeks on ‘day of week’” | PatternType = Weekly  PatternPeriod = N  PatternDOWMask = ‘Day of Week’ |
| “Repeat every weekday” | PatternType = Weekly  PatternPeriod = 1  PatternDOWMask = M, Tu, W, Th, F |
| “Repeat every weekend” | PatternType = Weekly  PatternPeriod = 1  PatternDOWMask = Su, Sa |

Table 3. Monthly Recurrence Patterns

|  |  |
| --- | --- |
| Patterns | Properties |
| “Repeat the lth day of every N months” | PatternType = Monthly  PatternPeriod = N  PatternDayOfMonth = J |
| “Repeat the Ith ‘day of week’ of every N months” | PatternType = Monthly  PatternPeriod = N  PatternDOWMask = ‘Day of Week’  PatternInstance = I |

Table 4. Yearly Recurrence Patterns

|  |  |
| --- | --- |
| Patterns | Properties |
| “Repeat every ‘month’ ‘day’” | PatternType = Yearly  PatternPeriod = 1  PatternDayOfMonth = ‘Day’  PatternMonthOfYear = ‘Month’ |
| “Repeat the Ith ‘day of week’ of ‘month’” | PatternType = Yearly  PatternPeriod = 1  PatternDOWMask = ‘Day of Week’  PatternInstance = I  PatternMonthOfYear = ‘Month’ |

#### 4.2.3.2 Moved Occurrences

You can implement support for moving occurrences of repeating appointments by using MTP object references and the AbstractActivityOccurrence format. Moved appointments might or might not contain changes to properties of the parent appointment. If a particular property is not specified on the moved occurrence, MTP device services use the value of the property in the parent appointment.

##### 4.2.3.2.1 Representing Moved Occurrences on the MTP Responder

The general model for representing a moved appointment at the MTP level is to do the following:

1. For each moved occurrence, create an object of format AbstractActivityOccurrence.

2. Specify the original date that the occurrence was to occur in the PatternOriginalTime property. Because this represents a specific instance in time, this property must be in UTC time.

3. Specify the new date and time, in UTC, for the appointment in the BeginTime property.

4. Update any additional properties based on the specific values for this occurrence. If the property has not changed from the parent appointment, you do not need to include the property on the moved occurrence.

5. Include the object ID for this moved occurrence in the list of object IDs that are returned by the GetObjectReferences operation that is associated with this object.

##### 4.2.3.2.2 Creating, Updating, and Deleting Occurrences Moved by the Initiator

When a user creates a new recurring item or modifies an existing recurring item, Windows does the following:

1. For each recurring item, first creates or updates the parent appointment by using a SendObjectPropList or SetObjectPropList operation. During this period, Windows does not call the SetObjectReferences operation.

2. For each occurrence of the recurring item, creates a new occurrence object. Windows does not yet call SetObjectReferences to associate the new occurrence item with the parent appointment.

3. After it creates all child objects, calls SetObjectReferences to set all moved occurrences.

Now the MTP responder must delete all references to any previously existing moved occurrences. The device must send appropriate MTP events as the old child objects are removed. The MTP responder is then responsible for associating the new set of object references with the parent appointment.

The process for deleting occurrences of a recurring appointment is similar to the process of creating or updating an occurrence in that the MTP responder is responsible for handling the actual deletion. To delete an occurrence of a recurring appointment, Windows does the following:

1. Uses GetObjectReferences to determine the current occurrence objects that a recurring appointment references.

2. Calls SetObjectReferences without the specified occurrence to remove the occurrence. To remove all occurrences, calls SetObjectReferences with an empty set.

As with create and update, the MTP responder must remove and update the contents of the object reference list and send all appropriate events.

The MTP responder should treat any attempt to delete an occurrence object that has a parent as an error.

## 4.3 Notes Service

The Notes service uses a synchronization service to synchronize notes to and from the device. Notes may contain HTML or plain formatted text.

### 4.3.1 Notes Service General Information

The general information for the Notes service includes the following:

* Service GUID: {5c017aea-e706-4719-8cc0-a303836fd321} (SVCGUID\_NOTES)
* Service Name: Notes
* Service Type: Default

### 4.3.2 Notes Service Properties

The Notes service supports the following service property namespace.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {703d392c-532c-4607-9158-9cea742f3a16} | SYNCSVC\_SERVICE\_PROPERTIES | 1 |

The Notes service supports the properties that it inherits from the Synchronization service.

### 4.3.3 Notes Service Formats

The Notes service supports the following service format.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GUID | Name | R/W | M/O | MIME type | Description |
| {b3d1b688-39f6-4703-b339-c69b7d2abb3f} | AbstractNote | RW | M | application/ octet-stream | An abstract note. |

The supported format is described in the following section.

#### 4.3.3.1 AbstractNote Format

The Notes service supports the following object property namespaces.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {ef6b490d-5cd8-437a-affc-da8b60ee4a3c} | Generic Object Properties | 1 |
| {B28AE94B-05A4-4E8E-BE01-72CC7E099D8F} | Information Object Properties | 3 |

##### 4.3.3.1.1 Properties

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 3 | ParentID | UINT32 | RO | M | Parent ID. |
| 1 | 4 | Name | STRING | DD | M | Display name for this object, which the device generates. We recommend the Subject property value. This property is shown to the user in error messaging. |
| 1 | 5 | PUOID | UINT128 | RO | M | Persistent object unique ID. This must be a GUID. |
| 1 | 6 | ObjectFormat | UINT16 | RO | M | MTP format code that this object represents. |
| 1 | 11 | ObjectSize | UINT64 | RO | O | Size of this object in bytes. Typically, note objects are of 0-byte length. |
| 1 | 23 | StorageID | UINT32 | RO | M | Storage ID for this object. |
| 1 | 28 | Hidden | UINT16 - ENUM | RO | O | Note objects should typically not be marked as hidden. In full enumeration synchronization operations, hidden note objects are not excluded from the synchronization. |
| 1 | 40 | DateModified | STRING – DateTime form | RW | O | Date on which this object was last modified. |
| 3 | 2 | Subject | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Title of the note. |
| 3 | 3 | Body | AUINT16 - LONGSTRING FORM (0xFF) | RW | M | Contents of the note. |
| 3 | 10 | Category | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of categories that apply to the note. |

##### 4.3.3.1.2 Stream Contents

No stream contents exist. This is an abstract (0-byte) object format.

## 4.4 Task Service

The Task service uses a synchronization service to synchronize tasks. Windows supports one Task service.

#### Property Groups

When specifying property group codes, the following properties must be in GroupCode 0x1:

* PercentComplete
* All properties in the VersionProps dataset from the Synchronization service

### 4.4.1 Task Service General Information

The general information for the Task service includes the following:

* Service GUID: {BB340C54-B5C6-491D-8827-28D0E7631903} (SVCGUID\_TASKS)
* Service Name: Tasks
* Service Type: Default

### 4.4.2 Task Service Properties

The Task service supports the following service property namespace.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {703d392c-532c-4607-9158-9cea742f3a16} | SYNCSVC\_SERVICE\_PROPERTIES | 1 |

The Task service supports the following property.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 4 | FilterType | UINT8 | RW | M | A property that is set by the synchronization host to track whether the content-type-specific default synchronization filter is applied. If a value is set, only active tasks are synchronized. |

### 4.4.3 Task Service Formats

The Task service supports the following service formats.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GUID | Name | R/W | M/O | MIME type | Description |
| {522979c0-74cf-44ab-9754-55bc596a67df} | AbstractTask | RW | M | application/ octet-stream | An abstract task. |
| {0xBE030000-AE6C-4804-98BA-C57B46965FE7} | iCalendar | RW | O | text/calendar | iCalendar file format (vCalendar Version 2). |

The following sections provide more information about the service formats.

#### 4.4.3.1 AbstractTask Format

The AbstractTask format supports the following property namespaces.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {ef6b490d-5cd8-437a-affc-da8b60ee4a3c} | Generic Object Properties | 1 |
| {E354E95E-D8A0-4637-A03A-0CB26838DBC7} | Task Service Object Properties | 2 |
| 0xB28AE94B, 0x05A4, 0x4E8E, 0xBE, 0x01, 0x72, 0xCC, 0x7E, 0x09, 0x9D, 0x8F | Information Object Properties | 3 |
| {0xF99EFD03-431D-40D8-A1C9-4E220D9C88D3} | Calendar Object Properties | 4 |

##### 4.4.3.1.1 Properties

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 3 | ParentID | UINT32 | RO | M | Parent ID. |
| 1 | 4 | Name | STRING | DD | M | Display name for this object, which the device generates. We recommend the Subject property value. This property is shown to the user in error messaging. |
| 1 | 5 | PUOID | UINT128 | RO | M | Persistent object unique ID. This must be a GUID. |
| 1 | 6 | ObjectFormat | UINT16 | RO | M | MTP format code that this object represents. |
| 1 | 11 | ObjectSize | UINT64 | RO | O | Size of this object in bytes. If this is an abstract task format, this is always 0. If iCalendar encoding is used, this is the length of the iCalendar encoding of the task. |
| 1 | 23 | StorageID | UINT32 | RO | M | Storage ID for this object. |
| 1 | 40 | DateModified | STRING – DateTime form | RW | O | Date on which this object was last modified. |
| 3 | 2 | Subject | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | M | Subject of the task. |
| 2 | 14 | Complete | UINT8 - ENUM | RW | M | Whether the task is complete or incomplete:   * 0x00 = Incomplete * 0xFF = Complete |
| 2 | 13 | ReminderDateTime | String – DateTime | RW | M | Time of the user to be alerted about the task. Date must comply with ISO8601 UTF16-LE. |
| 3 | 18 | Sender | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Owner of the task. |
| 3 | 10 | Category | STRING OR AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Semicolon-separated list of categories that apply to the task. |
| 3 | 4 | Priority | UINT16 - ENUM | RW | O | Priority of an object. This is usually attached to an e-mail or calendar/task item, but can be applied to any object for which the meaning can be inferred by the device. This property contains an integer that represents the identified priority of the object to which this property is attached. |
| 2 | 15 | BeginDate | STRING – ISO8601 Date | RW | O | Date on which work should begin on a task. The date is assumed to be relative to the current device time zone.  Date must comply with ISO8601 UTF16-LE. |
| 2 | 16 | EndDate | STRING – ISO8601 Date | RW | O | Date on which a task should be completed. The date is assumed to be relative to the current device time zone.  Date must comply with ISO8601 UTF16-LE. |
| 3 | 3 | Body | AUINT16 - LONGSTRING FORM (0xFF) | RW | O | Message or note that is associated with this task item. |
| 3 | 27 | PatternType | UInt32- ENUM | RW | M | Enumeration that indicates the type of the recurring appointment. Acceptable values are:   * 0x01 = Daily * 0x02 = Weekly * 0x03 = Monthly * 0x04 = Yearly |
| 3 | 28 | PatternValidStartDate | STRING – ISO8601 Date form | RW | M | Date on which the recurrence pattern becomes valid. All occurrences of the recurring appointment must occur on or after this date. |
| 3 | 29 | PatternValidEndDate | STRING – ISO8601 Date form | RW | M | Date on which the recurrence pattern is no longer valid. All occurrences of the recurring appointment must occur before or on this date. Only one of PatternValidEndDate and PatternValidCount may be set. |
| 3 | 31 | PatternPeriod | UInt32 | RW | M | Period for the repetition of the recurrence pattern. The units of this value is deter­mined by the recurrence type. If recurrence type is Daily, this is a count of days; if recurrence type is Weekly, this is a count of weeks; and so on. |
| 3 | 32 | PatternDayOfWeek | UInt16 | RW | M | For weekly, monthly, and yearly recurrence patterns, the day of week on which the pattern is valid. The field is a bitmask of the following values:   * 0x0000 = None * 0x0001 = Sunday * 0x0002 = Monday * 0x0004 = Tuesday * 0x0008 = Wednesday * 0x0010 = Thursday * 0x0020 = Friday * 0x0040 = Saturday |
| 3 | 33 | PatternDayOfMonth | UInt8 | RW | M | For monthly and yearly recurrence patterns, the day of the month on which the pattern is valid. The last day of the month is always 31, even on months that do not have 31 days. |
| 3 | 34 | PatternMonthOfYear | UInt8 | RW | M | For yearly recurrence patterns, the month of the year during which the pattern is valid. |
| 3 | 35 | PatternInstance | UInt8 | RW | M | For monthly and yearly recurrence patterns, the instance of the pattern that matches:   * 0x01 = the first instance * 0x02 = the second instance * 0x05 = the last instance |
| 3 | 36 | PatternDeleteDates | AUINT16 – LONGSTRING FORM | RW | M | A list of dates on which the recurrence pattern does not apply. This list contains only dates that have been deleted, not dates that have been moved. |
| 4 | 14 | TimeZone | STRING | RW | M | The time zone in which the appointment was created. Values for this field must come from the public-domain tz database (<http://www.twinsun.com/tz/tz-link.htm>). |
| 4 | 16 | BusyStatus | UInt16 | RW | O | How the appointment is to appear on the calendar:   * 0x0001 = Free * 0x0002 = Busy * 0x0003 = Out of Office * 0x0004 = Tentative |
| 4 | 17 | PatternStartTime | STRING – ISO8601 Time form | RW | M | The time of day, in local time, that the recurring appointment is to occur. |

##### 4.4.3.1.2 Stream Contents

If this object is an abstract task format, no data stream is associated with the object. If the format of the object is iCal, the data stream contains the iCalendar formatted data stream for this object.

During synchronization, only abstract properties for this object are updated. If the object format is iCalendar, it is the responsibility of the MTP responder to maintain synchronization between the iCal data stream and the abstract properties.

## 4.5 Status Service

A device Status service reports device status properties to the PC. When a user connects a device to a system that is running Windows, Windows queries the device for the initial state of all properties. When the value for a status property changes, the responder should send a ServicePropChanged event and the PC requeries that property. Table 5 lists the times at which the device should report changes in status. When you choose an update frequency, be sure to consider device and PC performance.

Table 5. Reporting Frequencies of Status Properties

|  |  |  |
| --- | --- | --- |
| Properties | Source | Reporting frequency |
| Signal Strength, Network Information | Status service | 1 minute |
| Battery | Status service | 1 minute |
| Text Messages | Status service | Instant (on receipt of message) |
| Missed Calls | Status service | Instant (on receipt of missed call) |
| Voicemail | Status service | Instant (on receipt of voice mail) |
| ChargingState | Status service | Instant (on change of charging status) |
| Pictures | Status service | At start of MTP session |
| Storage Space | Legacy storage Information | Provided by StorageInfo |

The properties that are described in the following sections are supported inbox by the Device Stage status pane. You can add additional properties to the device Status service. Device Stage uses the service property GUID to match the data values in the device service to display/formatting elements in the Device Stage status pane.

### 4.5.1 Status Service General Information

The general information for the Status service includes the following:

* Service GUID: {0B9F1048-B94B-DC9A-4ed7-fe4fed3a0deb} (SVCGUID\_STATUS)
* Service Name: Status
* Service Type: Default

### 4.5.2 Status Service Properties

The Status service supports the following service property namespace.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {49cd1f76-5626-4b17-a4e8-18b4aa1a2213} | STATUSSVC\_SERVICE\_PROPERTIES | 1 |

The Status service supports the following service properties.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 2 | SignalStrength | UINT8 - RANGE | RO | O | Signal strength, from 0 to 4. |
| 1 | 3 | TextMessages | UINT8 | RO | O | Number of unread text messages. |
| 1 | 4 | NewPictures | UINT16 | RO | O | Total number of pictures on the device. |
| 1 | 5 | MissedCalls | UINT8 | RO | O | Total number of missed calls on the device. |
| 1 | 6 | VoiceMail | UINT8 | RO | O | Total number of new voicemail messages on the device/service. For devices that have only a binary state, 0 represents no new voicemail messages and 0xFF represents new messages. |
| 1 | 7 | NetworkName | STRING | RO | O | Human-readable name of the current mobile network (for example, “Microsoft Cellular”). |
| 1 | 8 | NetworkType | STRING | RO | O | Type of mobile network that the device is currently using (for example, “E” for EDGE, “U” for UMTS, or “1x” for 1xRTT). |
| 1 | 9 | Roaming | UINT8 - ENUM | RO | O | * 0 if the mobile device is on its home network. * 1 if the device is roaming. * 2 if the roaming status is unknown. |
| 1 | 10 | BatteryLife | UINT8 - RANGE | RO | M | Remaining battery life of the device, as an integer from 0 to 100: |
| 1 | 11 | ChargingState | UINT8 - ENUM | RO | M | * 0 if the battery is not charging. * 1 if the battery is currently charging. * 2 if the charging status is unknown. |
| 1 | 12 | StorageCapacity | UINT64 | RO | M | Total usable storage capacity of the device, in bytes, across all storage locations. |
| 1 | 13 | StorageFreeSpace | UINT64 | RO | M | Total usable free space on the device, in bytes, across all storage locations. |
| 1 | 15 | InternetConnected | UINT8 - ENUM | RO | O | Boolean value that indicates whether the mobile device is connected to an outside data network (such as the Internet):   * 1 if it is connected. * 0 if it is not connected. |

## 4.6 Hints Service

The Hints service enables the device to select preferred storage locations for various content types in legacy storages.

The Hints service contains only data in the ServiceInfo dataset. In the Data section of the ServiceInfo dataset, list the WPD content types with a list of object IDs of folders for each content type. If a WPD content type has multiple ObjectIDs, Windows ranks the ObjectIDs in the order in which they are listed. The first ObjectID is the most preferred location for a particular content type, the second ObjectID is the second preferred location, and so on.

If the user deletes the hints locations (either specifically or by using a storage format), the device can re-create the folders either immediately or upon the next transfer. The ServiceInfo dataset should always contain the correct Object IDs of the hints locations. If the hints locations are changed, you should remove the service and add it back by using the ServiceRemoved and ServiceAdded events. You should re-create the hints locations rather than update the ServiceInfo.

### 4.6.1 Hints Service General Information

The general information for the Hints service includes the following:

* Service GUID: {c8a98b1f-6b19-4e79-a414-67ea4c39eec2} (SVCGUID\_Hints)
* Service Name: Hints
* Service Type: Default

The Hints service does not support any service properties, service formats, service methods, or service events.

#### 4.6.1.1 Data Stream

You should specify the data for this service in the Data section of the ServiceInfo dataset. The data should be in the following format.

|  |  |  |
| --- | --- | --- |
| Field order | Data  type | Description |
| 1 | UINT32 | Number of content-type location hints. |
| 2 | UINT128 | WPD content type GUID of the content for the hint. |
| 3 | UINT32 | Object handle of the parent object under which to store the specified content type. This object must be of type Association (0x3001). |

## 4.7 Device Metadata Service

The Windows Device Metadata service enables a device to deliver Device Stage metadata when a user connects the device to a Windows system. This service is faster than downloading Device Stage metadata from the Microsoft metadata servers, and it is the only way to obtain device metadata if no Internet connection exists at the time that the driver is installed. This service delivers the device metadata to the Device Metadata Retrieval Client (DMRC) store before display object processing.

For the specification that describes documents within the Windows CAB file, see the Microsoft Device Experience Development Kit on the WHDC Web site.

### 4.7.1 Device Metadata Service General Information

The general information for the Device Metadata service includes the following:

* Service GUID: {332ffe6a-af65-41e1-a0af-d3e2627bdf54} (SVCGUID\_Device\_Metadata\_Service)
* Service Name: MetaData
* Service Type: Default

The device metadata service has no service properties.

### 4.7.2 Device Metadata Service Formats

The Device Metadata service supports the following service format.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GUID | Name | R/W | M/O | MIME type | Description |
| {0xe1809599-4303-4e3b-9244-99c62c254551} | DeviceMetadataCAB | RO | M | application/ octet-stream | Metadata object |

The DeviceMetadataCAB format supports the following property namespaces.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {ef6b490d-5cd8-437a-affc-da8b60ee4a3c} | Generic Object Properties | 1 |
| { 68bb7eeb-9eef-45bd-8de6-3b92a57cae1e} | Device Metadata Service Namespace | 2 |

#### 4.7.2.1 Properties

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 3 | ParentID | UINT32 | RO | M | Parent ID. |
| 1 | 4 | Name | STRING | RW | M | Display name for this object. |
| 1 | 5 | PUOID | UINT128 | RO | M | Persistent object unique ID. This must be a GUID. |
| 1 | 6 | ObjectFormat | UINT16 | RO | M | MTP format code that this object represents. |
| 1 | 11 | ObjectSize | UINT64 | RO | O | Size of this object in bytes. |
| 1 | 23 | StorageID | UINT32 | RO | M | Storage ID for this object. |
| 1 | 27 | LanguageLocale | STRING | RO | M | Locale of the CAB contents. The locale must be composed of valid RFC4646 subtags (for example, “en-US”). |
| 2 | 3 | ContentID | UINT128 | RO | M | ID that uniquely identifies the CAB contents. This ID is a GUID that is assigned by the Windows logo signing process. |
| 2 | 4 | DefaultCAB | UINT8 - ENUM | RO | M | Boolean value that indicates whether the object is the default Device Metadata CAB object. The Device Metadata service must have only one object that is marked as default. |

## 4.8 Ringtone Service

A device stores ringtones without folder hierarchy on the target storage. Capabilities that are declared for the ringtone file formats can differ from capabilities for the media storage locations (bit rate, duration, and so on) and must be declared in the service. The ringtone service supports the WMA and MP3 formats. A device should support at least one format.

To avoid duplicate ringtones being sent to the device, the device must retain the Duration property as set by Windows. Devices should support multiple files of the same name in the ringtone service, because many ringtones can have common names such as “Ringtone1” or “Alarm2”.

To support associating a ringtone with a contact, contacts in the Contact service must support the Name property (which is used to display contacts) and the Ringtone property.

### 4.8.1 Ringtone Service General Information

The general information for the Ringtone service includes the following:

* Service GUID: {d0eace0e-707d-4106-8d38-4f560e6a9f8e} (SVCGUID\_Ringtones)
* Service Name: Ringtones
* Service Type: Default

### 4.8.2 Ringtone Service Properties

The Ringtone service supports the following service property namespace.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {7d05d925-32e6-4790-9205-54764bb3cb74} | RINGTONE\_SERVICE\_PROPERTIES | 1 |

The Ringtone service supports the following service property.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 2 | DefaultRingtone | UINT32 | RW | M | Object ID of default ringtone for incoming calls. |

### 4.8.3 Ringtone Service Formats

The Ringtone service supports the following service formats.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GUID | Name | R/W | M/O | MIME type | Description |
| 30090000-AE6C-4804-98BA-C57B46965FE7 | MP3File | RW | O | Audio/mp3 | MP3 file format |
| B9010000-AE6C-4804-98BA-C57B46965FE7 | WMAFile | RW | O | Audio/x-ms-wma | WMA file format |

The Ringtone service supports the following property namespaces.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {ef6b490d-5cd8-437a-affc-da8b60ee4a3c} | Generic Object Properties | 1 |
| B324F56A-DC5D-46E5-B6DF-D2EA414888C6 | Audio Properties | 2 |
| 2ED8BA05-0AD3-42DC-B0D0-BC95AC396AC8 | Media Properties | 3 |

The Ringtone service supports the following properties for all formats.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 3 | ParentID | UINT32 | RO | M | Parent ID. |
| 1 | 4 | Name | STRING | RW | M | Display name for this object. |
| 1 | 5 | PUOID | UINT128 | RO | M | Persistent object unique ID. This must be a GUID. |
| 1 | 6 | ObjectFormat | UINT16 | RO | M | MTP format code that this object represents. |
| 1 | 11 | ObjectSize | UINT64 | RO | O | Size of this object in bytes. |
| 1 | 40 | DateModified | STRING – DateTime Form | RW | M | Date on which this object was last modified. |
| 1 | 23 | StorageID | UINT32 | RO | M | Storage ID for this object. |
| 2 | 12 | AudioBitDepth | UINT32 | DD | O | Bit depth of the ringtone. For details, refer to the MTP Audio Bit Depth (0xDE95) object property. |
| 2 | 10 | Channels | UINT16 | DD | O | Number of audio channels for the ringtone. Windows outputs 1 or 2 channels. For details, refer to the MTP Number of Channels (0xDE94) object property. |
| 3 | 15 | SampleRate | UINT32 | DD | M | Sample rate of the ringtone. For details, refer to the MTP Sample Rate (0xDE93) object property. |
| 2 | 9 | AudioBitrate | UINT32 | DD | M | Bitrate of the ringtone. For details, refer to the MTP Audio Bitrate (0xDE9A) object property. |
| 2 | 11 | AudioFormatCode | UINT32 | DD | M | The WAVE code of the particular ringtone. For details, refer to MTP Audio WAVE Codec (0xDE99) object property. |
| 3 | 19 | Duration | UINT32 | RW | M | For details, refer to the MTP Duration (0xDC99) object property. |

## 4.9 Enumeration Synchronization Service

The Enumeration Synchronization service is a template service that other services can use. Enumeration-based synchronization is fully driven by Windows and requires limited support from an MTP responder. Synchronization occurs in two stages: enumeration of objects and updates. For more details on synchronization, see “[Synchronizing PIM Data](#_Synchronizing_PIM_Data)” later in this paper.

### 4.9.1 Enumeration Synchronization Service General Information

The general information for the Enumeration Synchronization service includes the following:

* Service GUID: {28d3aac9-c075-44be-8881-65f38d305909} (SVCGUID\_FullEnumSync)
* Service Name: FullEnumSync
* Service Type: Abstract

### 4.9.2 Enumeration Synchronization Service Properties

The Enumeration Synchronization service supports the following service property namespaces.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {63b10e6c-4f3a-456d-95cb-9894edec9fa5} | FULLENUMSYNC\_SERVICE\_PROPERTIES | 1 |
| {703d392c-532c-4607-9158-9cea742f3a16} | SYNCSVC\_SERVICE\_PROPERTIES | 2 |

The Enumeration Synchronization service supports the following service properties.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 3 | FullEnumVersionProps | UINT8 array | RO | M | Information about what properties on objects of type SyncFormat contain change-unit version IDs. For more information, see the paragraph that follows this table. |
| 1 | 4 | FullEnumReplicaID | UINT128 | RW | M | GUID that is set by the synchronization host at the first synchronization. |
| 1 | 7 | FullEnumKnowledgeObjectID | UINT32 | RO | O | ObjectID that should be used to read or write the anchor metadata for a partial participant. |
| 1 | 8 | FullEnumLastSyncProxyID | UINT128 | RW | M | A GUID that indicates the last synchronization proxy to perform a synchroniza­tion operation. This property is used for recovery and for handling synchronization software upgrades. |
| 1 | 9 | FullEnumProviderVersion | UINT16 | RO | M | A device-defined value that gives the version of the provider that is currently in use on the device. This version must be incremented when­ever new properties are added to the device implementa­tion so that they are recognized and managed as part of synchronization. The 0 value is reserved. |
| 2 | 2 | SyncFormat | UINT128 | RO | M | Format GUID for the objects that are to be synchronized. |
| 2 | 3 | LocalOnlyDelete | UINT8 - ENUM | RW | M | Boolean that is set by the synchronization host to indicate whether dele­tions on this endpoint propa­gate to other endpoints. This value is set by the PC and stored by the device. The device should not alter this value. |
| 2 | 4 | FilterType | UINT8 | RW | M | A value that is set by the synchronization host to indicate whether the content-type-specific default synchronization filter is applied. This value is set by the PC and carried by the device. |
| 2 | 5 | SyncObjectReferences | UINT8 – Enum | RO | O | Boolean value that describes whether object references should be included as part of the synchronization process:   * 0x0 = References Disabled * 0xFF = References Enabled |

#### 4.9.2.1 VersionProps

The VersionProps property contains information about what properties on objects of type SyncFormat contain change-unit version IDs. The Enumeration Synchronization service client retrieves the stream to get information about which change units are supported on the device. If the device supports only object-level changes, it must specify one change unit ID: GUID\_Null, 0. The stream is formatted as shown in Table 6.

Table 6. Enumeration Synchronization Service Stream Format

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Change Unit Count | UINT32 | Number of change-unit entries in the stream. |
| Change Unit 1 | PKEY (UINT128, UINT32) | ID of the first change unit. If only object-level changes are supported, this must be the “GUID\_Null, 0” PKey. |
| Version Property Count for Change Unit 1 | UINT32 | Number of properties that are associated with this change unit. |
| Version Property 1 for Change Unit 1 | PKEY (UINT128, UINT32) | First version property for this change unit. |
| … |  |  |
| Version Property N for Change Unit 1 | PKEY (UINT128, UINT32) | Last version property for this change unit. |
| Change Unit 2 | PKEY (UINT128, UINT32) | ID of the second change unit. |
| Version Property Count for Change Unit 2 | UINT32 | Number of properties that are associated with the version information for this change unit. |
| Version Property 1 for Change Unit 2 | PKEY (UINT128, UINT32) | First version property for this change unit. |
| … |  |  |

### 4.9.3 Enumeration Synchronization Service Formats

The Enumeration Synchronization service supports the following format.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GUID | Name | R/W | M/O | MIME type | Description |
| { 221bce32-221b-4f45-b48b-80de9a93a44a} | FullEnumSyncKnowledge | RW | O | application/ octet-stream | The metadata store that is associated with this synchronization replica. It is set by the synchronization host and is opaque to the device. |

#### 4.9.3.1 FullEnumSyncKnowledge Format

The FullEnumSyncKnowledge format supports the following property namespace.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {ef6b490d-5cd8-437a-affc-da8b60ee4a3c} | Generic Object Properties | 1 |

##### 4.9.3.1.1 Properties

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 3 | ParentID | UINT32 | RO | M | Parent ID of this object. All knowledge objects must be at the root, so this must always be set to 0x00000000. |
| 1 | 4 | Name | STRING | RW | M | Display name for this object. |
| 1 | 5 | PUOID | UINT128 | RO | M | Persistent object unique ID. This must be a GUID. |
| 1 | 6 | ObjectFormat | UINT16 | RO | M | MTP format code that this object represents. |
| 1 | 11 | ObjectSize | UINT64 | RO | M | Size of this object in bytes. |
| 1 | 23 | StorageID | UINT32 | RO | M | Storage ID for this object. |
| 1 | 28 | Hidden | UINT16 - ENUM | RO | M | Knowledge objects must be marked as hidden. This value must be set to 0x0001. |
| 1 | 13 | NonConsumable | UINT8 | RO | O | Nonconsumable flag. The anchor data is not meant for device consumption, so this property must always be 0x01 if the property is present. |
| 1 | 40 | DateModified | STRING – DateTime Form | RW | M | Date on which this object was last modified. |

##### 4.9.3.1.2 Stream Contents

This stream contains opaque data that the synchronization host manages. The device is responsible for persisting and retrieving this data as requested by the host, but it must *not* modify the data.

### 4.9.4 Enumeration Synchronization Service Methods

The FullEnumSyncKnowledge format supports the following service methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GUID | Name | Associated object | M/O | Description |
| {63803e07-c713-45d3-8119-3479b31d3592} | BeginSync | Anchor synchronization service | O | Locks device to begin a synchronization operation. |
| {40f3f0f7-a539-422e-98dd-fd8d385c8849} | EndSync | Anchor synchronization service | O | Unlocks device at the end of a synchronization operation. |

The following sections describe the methods.

#### 4.9.4.1 BeginSync Method

BeginSync is an optional method that can be supported by services that implement the FullEnumSync service behaviors. Devices that require locking semantics, such as to prevent user-driven changes on the device, can support the BeginSync method and use it to indicate when synchronization is in progress.

BeginSync takes no parameters and returns no parameters.

#### 4.9.4.2 EndSync Method

EndSync is an optional method that can be supported by services that implement the FullEnumSync service behaviors. If an MTP responder supports BeginSync on the service, it must also support EndSync. MTP responders can use the EndSync method to remove any locking semantics that might have been created by calling the BeginSync method.

EndSync takes no parameters and returns no parameters.

## 4.10 Anchor Synchronization Service

The Anchor Synchronization service is an abstract MTP device service that is defined to enable other services to opt in to a synchronization relationship by using anchor synchronization semantics. Because this service is an abstract service, the actual implementation of the properties, formats, and methods that are defined in the service is the responsibility of the “concrete” service that uses them.

Devices that support device synchronization with Windows 7 components should use the Anchor Synchronization service, instead of the full Enumeration Synchronization service, to achieve optimal performance. The anchor functionality places additional demands on the device to be able to return anchor-specific information about items that have been created, updated, or deleted. If a device cannot return this information, it must use the full Enumeration Synchronization service instead.

For more information on synchronization, see “[Synchronizing PIM Data](#_Synchronizing_PIM_Data)” later in this paper.

### 4.10.1 Anchor Synchronization Service General Information

The general information for the Anchor Synchronization service includes the following:

* Service GUID: {056d8b9e-ad7a-44fc-946f-1d63a25cda9a} (SVCGUID\_AnchorSync)
* Service Name: AnchorSync
* Service Type: Abstract

### 4.10.2 Anchor Synchronization Service Properties

The Anchor Synchronization service supports the following service property namespaces.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {e65b8fb7-8fc7-4278-b9a3-ba14c2db40fa} | ANCHORSYNCSVC\_SERVICE\_PROPERTIES | 1 |
| {703d392c-532c-4607-9158-9cea742f3a16} | SYNCSVC\_SERVICE\_PROPERTIES | 2 |

The Anchor Synchronization service supports the following properties.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 2 | AnchorVersionProps | UINT8 array | RO | M | Information about which properties for objects of type SyncFormat contain change-unit version IDs. For more information, see the para­graph that follows this table. |
| 1 | 3 | AnchorReplicaID | UINT128 | RW | M | GUID that is set by the synchronization host at the first synchronization. |
| 1 | 4 | AnchorKnowledgeObjectID | UINT32 | RO | M | ObjectID that should be used to read or write the anchor metadata for a partial participant. |
| 1 | 5 | AnchorLastSyncProxyID | UINT128 | RW | M | A GUID that indicates the last synchronization proxy to perform a synchronization operation. This property is used for recovery and for handling synchronization software upgrades. |
| 1 | 6 | AnchorCurrentAnchor | AUINT8 binary array | RW | M | A block of data that repre­sents the current anchor for the device. The anchor might be transient. Depending on the current state of the synchronization, the value of PKEY\_AnchorSyncSvc \_CurrentAnchor may not reflect the current state of the database unless the current session holds a lock (through the BeginSync method) on the service. |
| 1 | 7 | AnchorProviderVersion | UINT | RO | M | A device-defined value for the version of the provider that is currently used on the device. This version must be incre­mented whenever new properties are added to the device implementation so that they are recognized and managed as part of synchron­i­zation. The value 0 is reserved. |
| **2** | 2 | SyncFormat | UINT128 | RO | M | Format GUID for the objects that are to be synchronized. |
| **2** | 3 | LocalOnlyDelete | UINT8 - ENUM | RW | M | Boolean that is set by the synchronization host to indicate whether deletions on this endpoint propagate to other endpoints. |
| 2 | 4 | FilterType | UINT8 | RW | M | A type that is set by the synchronization host to indicate whether the content-type-specific default synchronization filter is applied. |
| 2 | 5 | SyncObjectReferences | UINT8 – ENUM | RO | O | Boolean value that describes whether object references should be included as part of the synchronization process:   * 0x0 = References Disabled * 0xFF = References Enabled |

#### 4.10.2.1 AnchorVersionProps

The AnchorVersionProps property contains information about which properties for objects of type SyncFormat contain change-unit version IDs. The Anchor service client retrieves the stream to get information about which change-units are supported on the device. If the device supports only object-level changes, it must specify one changeunit ID: GUID\_Null, 0. The stream is formatted as shown in the following table.

#### 4.10.2.2 Stream Format

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Change Unit Count | UINT32 | Number of change-unit entries that follow. |
| Change Unit 1 | PKEY (UINT128, UINT32) | ID of the first change unit. If only object-level changes are supported, this must be the “GUID\_Null, 0” PKey. |
| Version Property Count for Change Unit 1 | UINT32 | Number of properties that are associated with this change unit. |
| Version Property 1 for Change Unit 1 | PKEY (UINT128, UINT32) | First version property for this change unit. |
| … |  |  |
| Version Property N for Change Unit 1 | PKEY (UINT128, UINT32) | Last version property for this change unit. |
| Change Unit 2 | PKEY (UINT128, UINT32) | ID of the second change unit. |
| Version Property Count for Change Unit 2 | UINT32 | Number of properties that are associated with the version information for this change unit. |
| Version Property 1 for Change Unit 2 | PKEY (UINT128, UINT32) | First version property for this change unit. |
| … |  |  |

### 4.10.3 Anchor Synchronization Service Formats

The Anchor Synchronization service supports the following service formats.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| GUID | Name | R/W | M/O | MIME type | Description |
| { 37c550bc-f231-4727-bbbc-4cb33a3f3ecd} | AnchorKnowledge | RW | M | application/ octet-stream | The metadata store that is associated with this synchroni­­zation replica. It is set by the synchronization host and is opaque to the device. For more information, see "[Anchor Knowledge Format](#_4.10.3.1__)" below. |
| {f35527c1-ce4a-487a-9d29-93833569321e} | AnchorResults | RO | M | application/ octet-stream | A service format that is created by the GetChangesSinceAnchor method. For more information, see "[Anchor Results Format](#_4.10.3.2__)" below. |

The following sections provide more information about the service formats.

#### 4.10.3.1 Anchor Knowledge Format

The Anchor Knowledge format supports the following property namespaces.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {ef6b490d-5cd8-437a-affc-da8b60ee4a3c} | Generic Object Properties | 1 |

##### 4.10.3.1.1 Properties

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 3 | ParentID | UINT32 | RO | M | Parent ID of this object. All AnchorKnowledge objects must be at the root, so this must always be set to 0x00000000. |
| 1 | 4 | Name | STRING | RW | M | Display name for this object. |
| 1 | 5 | PUOID | UINT128 | RO | M | Persistent object unique ID. This must be a GUID. |
| 1 | 6 | ObjectFormat | UINT16 | RO | M | MTP format code that this object represents. |
| 1 | 11 | ObjectSize | UINT64 | RO | M | Size of this object in bytes. |
| 1 | 23 | StorageID | UINT32 | RO | M | Storage ID for this object. |
| 1 | 28 | Hidden | UINT16 - ENUM | RO | M | Knowledge objects must be marked as hidden. This value must be set to 0x0001. |
| 1 | 30 | NonConsumable | UINT8 | RO | O | Nonconsumable flag. The anchor data is not meant for device consumption, so this property must always be 0x01 if the property is present. |
| 1 | 40 | DateModified | STRING - DateTime | RW | M | Date on which this object was last modified. |

##### 4.10.3.1.2 Stream Contents

This stream contains opaque data that is managed by the synchronization host. The device is responsible for persisting and retrieving this data as requested by the host, but must *not* modify the data.

#### 4.10.3.2 Anchor Results Format

The Anchor Results format supports the following property namespace.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| {ef6b490d-5cd8-437a-affc-da8b60ee4a3c} | Generic Object Properties | 1 |

##### 4.10.3.2.1 Properties

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 1 | 3 | ParentID | UINT32 | RO | M | Parent ID of this object. All anchor results objects must be at the root, so this must always be set to 0x00000000. |
| 1 | 4 | Name | STRING | RW | M | Display name for this object. |
| 1 | 5 | PUOID | UINT128 | RO | M | Persistent object unique ID. This must be a GUID. |
| 1 | 6 | ObjectFormat | UINT16 | RO | M | MTP format code that this object represents. |
| 1 | 11 | ObjectSize | UINT64 | RO | M | Size of this object in bytes. |
| 1 | 23 | StorageID | UINT32 | RO | M | Storage ID for this object. |
| 1 | 28 | Hidden | UINT16 - ENUM | RO | M | Anchor results objects must be marked as hidden. This value must be set to 0x0001. |
| 1 | 30 | NonConsumable | UINT8 | RO | O | Nonconsumable flag. The anchor data is not meant for device consumption, so this property must always be 0x01 if present. |
| 1 | 40 | DateModified | STRING - DateTime | RW | O | Date on which this object was last modified. |
| 1 | 26 | ProtectionStatus | UINT16 | RO | M | Results objects that may be read and deleted but not modified. This value must always be 0x0001. |

##### 4.10.3.2.2 Stream Contents

This stream is created in response to calling the GetChangesSinceAnchor method and contains information about all changes that occurred on the device since the specified anchor. The data is formatted into the stream as shown in the following table.

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Number of changes | UINT32 | Number of changes in this dataset. |
| Persistent Unique Object ID 1 | UINT128 | PUOID of the first object that changed. |
| Change Flags 1 | UINT32 | Change flags that are associated with this item. This UINT32 must contain one of the following values:   * 0x00000001 – Item was deleted * 0x00000002 – Item was created * 0x00000004 – Item was updated * 0x00000008 – Item was created or updated |
| Persistent Unique Object ID 2 | UINT128 | PUOID of the next changed object. |
| Change Flags 2 | UINT32 | Change flags for the next changed object. |
| … |  |  |
| Anchor Size | UINT32 | Size of the anchor when this result was generated. |
| Anchor | UINT8 Array | Value of the anchor when this result was generated. The anchor is opaque to the synchronization host. |

### 4.10.4 Anchor Synchronization Service Methods

The Anchor Synchronization service supports the following service methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GUID | Name | Associated object | M/O | Description |
| { 37c550bc-f231-4727-bbbc-4cb33a3f3ecd} | GetChangesSinceAnchor | Anchor Synchronization service | M | Returns details about all changes since the specified anchor. |
| {63803e07-c713-45d3-8119-3479b31d3592} | BeginSync | Anchor Synchronization service | O | Locks device to begin a synchronization operation. |
| {40f3f0f7-a539-422e-98dd-fd8d385c8849} | EndSync | Anchor Synchronization service | O | Unlocks device at the end of a synchroniza­tion operation. |

The following sections provide additional information about the supported service methods.

#### 4.10.4.1 GetChangesSinceAnchor Method

The GetChangesSinceAnchor method supports the following property namespaces.

|  |  |  |
| --- | --- | --- |
| GUID | Name | Namespace |
| { 516b5dce-8d45-430f-805c-25e5106d8b1f} | GetChangesSinceAnchor Parameters | 1 |
| {ef6b490d-5cd8-437a-affc-da8b60ee4a3c} | Generic Object Properties | 2 |

##### 4.10.4.1.1 Method Object Properties

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | R/W | M/O | Description |
| 2 | 3 | ParentID | UINT32 | RO | M | Parent ID of this object. All GetChangesSinceAnchor property objects must be at the root, so this must always be set to 0x00000000. |
| 2 | 6 | ObjectFormat | UINT16 | RO | O | MTP format code that this object represents. |
| 2 | 28 | Hidden | UINT16 - ENUM | RO | M | Knowledge objects must be marked as hidden. This value must be set to 0x0001. |
| 2 | 40 | DateModified | STRING - DateTime | RO | O | Timestamp that indicates when the method completed. |
| 2 | 23 | StorageID | UINT32 | RW | M | Storage ID for this object. |
| 2 | 26 | ProtectionStatus | UINT16 | RO | M | Method objects that can be read and deleted but not modified. This value must always be 0x0001. |

Methods do not require a persistent unique object ID because method objects do not persist.

##### 4.10.4.1.2 Method Input Parameters

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | Param-eter | R/W | M/O | Description |
| 1 | 3 | Anchor | AUINT8 | 1 | WO | M | The anchor after which changes should be reported. This value must be a value that the device returned, or it must be a zero-byte array that indicates “from the beginning”. |

##### 4.10.4.1.3 Method Output Parameters

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name-space | ID | Name | Type | Param-eter | R/W | M/O | Description |
| 1 | 2 | AnchorState | UINT32 | 0 | RO | M | Current state of the anchor data:   * 0 – Anchor is valid. * 1 – Anchor is too old or invalid |
| 1 | 4 | AnchorResults | UINT32 - Object ID | 1 | RO | M | Anchor after which changes should be reported. This value must be a value that the device returned, or it must be a zero-byte array that indicates “from the beginning”. |

#### 4.10.4.2 BeginSync Method

BeginSync is an optional method that can be supported by services that implement the FullEnumSync service behaviors. Devices that require locking semantics, such as to prevent user-driven changes on the device, can support the BeginSync method and use it to indicate when synchronization is in progress.

BeginSync takes no properties and returns no properties.

#### 4.10.4.3 EndSync Method

EndSync is an optional method that can be supported by services that implement the FullEnumSync service behaviors. If an MTP responder supports BeginSync on the service, it must also support EndSync. MTP responders can use the EndSync method to remove any locking semantics that might have been created by calling the BeginSync method.

EndSync takes no properties and returns no properties.

# 5.0 Synchronizing PIM Data

Synchronizing PIM data in Windows 7 requires combining a PIM data service with a synchronization service. A PIM service, such as Contacts, uses the Uses field in the ServiceInfo dataset to indicate which Synchronization services it supports.

## 5.1 Selecting a Synchronization Service

Windows 7 supports two synchronization services: Enumeration Synchronization and Anchor Synchronization. Enumeration synchronization is better used for devices that have no native synchronization support and rely on the PC for all change comparison work. Anchor synchronization is better for devices that can monitor changes since the last synchronization. Devices that support SyncML are better suited for anchor synchronization. Although the initial synchronization time is comparable for both methods, subsequent synchronizations may be significantly faster for anchor synchronization implementations. Because of the potential performance improvements, we recommend anchor synchronization for devices that support the required logic.

## 5.2 Enumeration Synchronization

Enumeration synchronization relies on simple comparison methods and requires no native support for synchronization on the device. As the name implies, Windows simply enumerates objects in a PIM service, compares each object with the PC’s data store, and updates each side accordingly. Windows stores a data record object, which is referred to as the knowledge object, for each PIM service that is paired to synchronization. The device is not required to consume this object and must simply persist it. Removal of the knowledge object resets the synchronization history and relationship for that particular PIM service.

## 5.3 Anchor Synchronization

The second type of enumeration that a device might support requires the use of an ”anchor”—a set of bytes that the device can use to identify how the synchronized data has changed. Anchors are meant only for the device that created them. Because only the device must know what the anchor means, a device or provider can choose whatever anchor it wants. Some anchors are based on generation version ticks of the database. Others might be based on the last modified time of an object.

## 5.4 More Information on PIM Synchronization

For information about writing synchronization plug-ins, choosing synchronization services, and PIM schema, see the document titled "Writing a Synchronization Provider for Windows Device Stage Synchronization" in the Portable Devices section of the Microsoft Device Experience Development Kit.

# 6.0 Resources

USB Media Transfer Protocol Specification, Revision 1.0

<http://www.usb.org/developers/devclass_docs/MTP_1.0.zip>

Microsoft Device Experience Development Kit

<http://www.microsoft.com/whdc/device/DeviceExperience/Dev-Kit.mspx>

MTP Device Services Extension Specification

<http://www.microsoft.com/whdc/device/wpd/MTPDevServExt_Spec.mspx>

Windows Portable Devices

<http://www.microsoft.com/whdc/device/wpd/default.mspx>

Windows Portable Devices Team Blog

<http://blogs.msdn.com/wpdblog/default.aspx>

WPD Application Programming Interface

<http://msdn.microsoft.com/en-us/library/dd389005(VS.85).aspx>