[MS-DCLB]:
Desktop Clipboard Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
08/14/2009	0.1	Major	First Release.
09/25/2009	0.1.1	Editorial	Revised and edited the technical content.
11/06/2009	0.1.2	Editorial	Revised and edited the technical content.
12/18/2009	0.1.3	Editorial	Revised and edited the technical content.
01/29/2010	0.2	Minor	Updated the technical content.
03/12/2010	0.2.1	Editorial	Revised and edited the technical content.
04/23/2010	0.2.2	Editorial	Revised and edited the technical content.
06/04/2010	0.3	Minor	Updated the technical content.
07/16/2010	0.3	No change	No changes to the meaning, language, or formatting of the technical content.
08/27/2010	0.3	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	0.3	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	0.3	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	0.3	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	0.3	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This is a specification of the Desktop Clipboard Protocol, which uses the **Network Dynamic Data Exchange (NetDDE)** Protocol to implement a distributed store for graphical user interface (GUI) objects for desktop cut-and-paste operations. It specifies the mechanism by which the Windows ClipBook Viewer application (the Microsoft Windows® **clipboard**) communicates information between remote users.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

American National Standards Institute (ANSI) character set
ASCII
big-endian
client
enhanced metafile format (EMF)
little-endian
metafile
NetBIOS
server
Unicode
Windows metafile format (WMF)

The following terms are specific to this document:

bitmap: A collection of structures that contain a representation of a graphical image, a **palette**, dimensions, and other information.

clipboard: A set of functions and messages that enable applications to transfer data. Transfers between applications are typically accomplished via "cut", "copy", or "paste" operations using the system **clipboard** implementation.

clipboard format: An identifier for the type of data that is stored in the **clipboard**. The **clipboard** can store multiple **clipboard formats** simultaneously.

clipbook: Clipboard data that is stored separately from the system **clipboard**.

color plane: One of the dimensions of a **color space**.

color space: A mapping of color components to a multidimensional coordinate system. The number of dimensions is generally two, three, or four. For example, if colors are expressed as a combination of the three components red, green, and blue, a three-dimensional space is sufficient to describe all possible colors.

device-independent bitmap (DIB): A container for **bitmapped** graphics, which specifies characteristics of the **bitmap** such that it can be created using one application and loaded and displayed in another application, while retaining an identical appearance.

dynamic data exchange (DDE): A protocol through which applications can exchange messages and use shared memory to exchange data. Applications can use **DDE** for one-time data transfers and for continuous exchanges in which applications send updates to each other as new data becomes available.

intensity: The magnitude of a component color in the **color space**.

logical palette: A **palette** that defines colors as device-independent values. Unlike the **system palette**, which has predefined, device-specific color definitions, a **logical palette** contains color values that can be defined entirely by an application. A **logical palette** entry must be mapped to the **system palette** entry in order for the custom colors to appear when the application is run.

mapping mode: The way in which logical (device-independent) coordinates are mapped to device-specific coordinates.

METAFILEPICT: A structure that defines the **metafile** picture format. **METAFILEPICT** is used for exchanging **metafile** data through the **clipboard**. See [MSDN-METAFILEPICT] and [MSDN-CLIPFORM] for further information.

Network Dynamic Data Exchange (NetDDE): A technology that allows applications using **dynamic data exchange (DDE)** to transparently share data over a network.

palette: An array of values, each element of which contains the definition of a color. The color elements in a **palette** are often indexed so that **clients** can refer to the colors, each of which can occupy 24 bits or more, by a number that requires less storage space.

system palette: The **palette** that is actually in use to reproduce colors on a device such as a computer screen. A **system palette** has predefined, device-specific colors that are used by default, so that every application does not have to set them up.

Tag Image File Format (TIFF): A format for **bitmapped** image data that comes from scanners, frame grabbers, and photo-retouching applications. It supports the exchange of image data between applications, taking advantage of the varying capabilities of imaging devices. **TIFF** supports a number of compression schemes that allow the choice of the best space or time tradeoff for applications.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-DTYP] Microsoft Corporation, "Windows Data Types", January 2007.

[ISO/IEC-8859-1] International Organization for Standardization, "Information Technology -- 8-Bit Single-Byte Coded Graphic Character Sets -- Part 1: Latin Alphabet No. 1", ISO/IEC 8859-1, 1998, http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=28245&ICS1=35&ICS2=40&ICS3=

Note There is a charge to download the specification.

[MS-EMF] Microsoft Corporation, "Enhanced Metafile Format", July 2007.

[MS-WMF] Microsoft Corporation, "Windows Metafile Format", June 2007.

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.ietf.org/rfc/rfc5234.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MSDN-CLIPBOARD] Microsoft Corporation, "Clipboard", http://msdn.microsoft.com/en-us/library/ms648709(VS.85).aspx

[MSDN-CLIPFORM] Microsoft Corporation, "Clipboard Formats", http://msdn.microsoft.com/en-us/library/ms649013.aspx

[MSDN-META] Microsoft Corporation, "Metafiles", http://msdn.microsoft.com/en-us/library/dd145051(VS.85).aspx

[MSDN-METAFILEPICT] Microsoft Corporation, "METAFILEPICT Structure", http://msdn.microsoft.com/en-us/library/ms649017(VS.85).aspx

[MSDN-NETDDE] Microsoft Corporation, "Network Dynamic Data Exchange", http://msdn.microsoft.com/en-us/library/aa365778(VS.85).aspx

1.3 Overview

The Desktop Clipboard Protocol is used by the Microsoft Windows® ClipBook Viewer application [MSDN-CLIPBOARD]. This protocol allows for transfer of clipboard data such as text and **bitmap** images between two participating machines.

In this protocol, the **server** role is defined as the actor that shares its clipboard data and provides this data when requested by the **client**. The client role is the actor that requests clipboard data from the server and is able to display this data to the user.

A sequence diagram showing this relationship is presented in section 3.

1.4 Relationship to Other Protocols

The Desktop Clipboard Protocol is implemented on top of the NetDDE protocol, as shown in the following diagram.

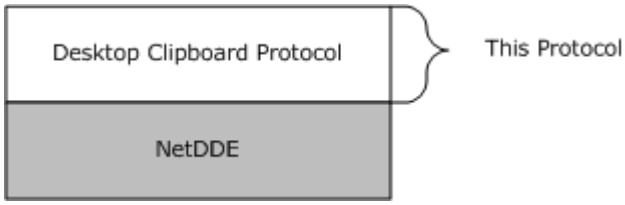


Figure 1: Relationship to other protocols

1.5 Prerequisites/Preconditions

The Desktop Clipboard Protocol requires the NetDDE API [MSDN-NETDDE].

1.6 Applicability Statement

The Desktop Clipboard Protocol is used for transferring clipboard information between remote machines.

1.7 Versioning and Capability Negotiation

The Desktop Clipboard Protocol does not have multiple versions.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

This protocol references commonly used data types as defined in [MS-DTYP].

2.1 Transport

The NetDDE API is used to initiate and maintain the network connections used by this protocol.

Desktop Clipboard messages are encapsulated in data blocks delivered using NetDDE as specified in section 3.1. The NetDDE transport ensures that the total message size is known.

2.2 Message Syntax

Note All unsigned 16-bit and unsigned 32-bit values are specified in **little-endian** format. Depending on the hardware architectures of the client and the server, multiple-byte little-endian versus **big-endian** reordering can determine how values are marshaled by the sender and interpreted by the receiver.

2.2.1 Common Field Values

2.2.1.1 ClipboardFormatName

The **ClipboardFormatName** constants are null-terminated **ANSI** strings ([ISO/IEC-8859-1]) that specify the **clipboard format**.

Constant/value	Description
CF_BITMAP "&Bitmap"	A handle to a bitmap.
CF_DIB "&DIB Bitmap"	A memory object containing a **device-independent bitmap (DIB)** structure ([MS-WMF] section 2.2.2.9), which consists of an information header followed by the bitmap bits.
CF_DIF "&DIF"	Software Arts' Data Interchange Format.
CF_DSPTEXT "Disp&lay Text"	Text display format associated with a private format.
CF_DSPBITMAP "Displa&y Bitmap"	Bitmap display format associated with a private format.
CF_DSPENHMETAFILE "Display En&hanced Metafile"	**EMF** display format associated with a private format.
CF_DSPMETAFILEPICT "Display Pict&ure"	**Metafile** picture display format associated with a private format.
CF_ENHMETAFILE "&Enhanced Metafile"	A handle to an EMF metafile [MS-EMF].
CF_METAFILEPICT	A handle to a metafile picture format.

Constant/value	Description
"&Picture"	
CF_OEMTEXT "&OEM Text"	Text format containing characters in the OEM character set. Each line ends with a CR/LF combination. A null character signals the end of the data.
CF_PALETTE "Pal&ette"	A handle to a color **palette**.
CF_PENDATA "Pe&n Data"	Data for the pen extensions to the Microsoft Windows for Pen Computing.
CF_RIFF "&RIFF"	Represents audio data that is more complex than can be represented in a CF_WAVE standard wave format.
CF_SYLK "&Sylk"	Microsoft Symbolic Link (SYLK) format.
CF_TEXT "&Text"	Text format. Each line ends with a carriage return/linefeed (CR/LF) combination. A null character signals the end of the data. Use this format for ANSI text.
CF_TIFF "T&IFF"	**Tag Image File Format (TIFF)**.
CF_UNICODETEXT "&Unicode Text"	**Unicode** text format. Each line ends with a CR/LF combination. A null character signals the end of the data.
CF_WAVE "&Wave Audio"	Represents audio data in one of the standard wave formats, such as 11 kHz or 22 kHz Pulse Code Modulation (PCM).

2.2.1.2 DDETopicType

The **DDETopicType** constants are character strings that specify the string constants used in **dynamic data exchange (DDE)** function calls.

Constant/value	Description
SZDDESYS_ITEM_TOPICS "Topics"	A list of the topics that are supported by the server at the current time.
SZDDESYS_TOPIC "System"	The system topic.

2.2.1.3 ExecuteCommandType

The **ExecuteCommandType** enumeration specifies the type of command message sent between actors in a **clipbook**-sharing session.

Constant/value	Description
CMD_DELETE "[delete]"	The sender is directing the recipient to remove a clipbook entry from its set of clipboard shares.

Constant/value	Description
CMD_INITSHARE "[initshare]"	The recipient has been requested to initialize its list of clipbook entries intended for sharing.
CMD_PASTE "[paste]"	The sender is directing the recipient to create a new clipbook entry.
CMD_SHARE "[markshared]"	The sender is directing the recipient to share a clipbook entry.
CMD_UNSHARE "[markunshared]"	The sender is directing the recipient to stop sharing a clipbook entry.

2.2.1.4 Notifications

The **Notification** constants are used by clients to request server action using the server's **DdeCallback** function.

Constant/value	Description
XTYP_ADVREQ 0x2022	Informs the server that a data request is outstanding on the specified topic name and item name pair, and that data corresponding to the pair has changed.
XTYP_REQUEST 0x20B0	Requests data from the server.
XTYP_EXECUTE 0x4050	Sends a command string to the server.

2.2.1.5 PaletteEntryFlags

The **PaletteEntryFlags** constants are used to describe palette data sent between users in a clipbook-sharing session.

Constant/value	Description
PC_DEFAULT 0x0000	Specifies default behavior for the **logical palette** entry.
PC_RESERVED 0x0001	Specifies that the logical palette entry be used for palette animation. This flag prevents other windows from matching colors to the palette entry since the color frequently changes. If an unused **system palette** entry is available, the color is placed in that entry. Otherwise, the color is not available for animation.
PC_EXPLICIT 0x0002	Specifies that the low-order word of the logical palette entry designates a hardware palette index. This flag allows the application to show the contents of the display device palette.
PC_NOCOLLAPSE 0x0004	Specifies that the color be placed in an unused entry in the system palette instead of being matched to an existing color in the system palette. If there are no unused entries in the system palette, the color is matched normally. Once this color is in the system palette, colors in other logical palettes can be matched to this color.

2.2.1.6 SharingStatusType

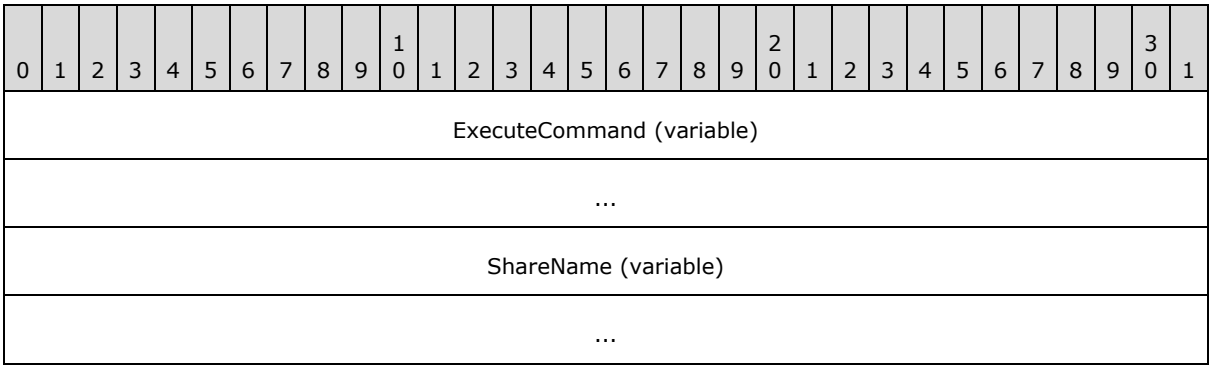
The **SharingStatusType** constants are ANSI characters that specify the sharing status of a clipbook.

Constant/value	Description
STATUS_SHARED "$"	The clipbook is shared.
STATUS_UNSHARED "*"	The clipbook is not shared.
STATUS_UPDATED "?"	The clipbook sharing status has been updated.

2.2.2 Control Information

2.2.2.1 EXECCOMMAND

The EXECCOMMAND structure specifies state changes for the clipbook-sharing session.

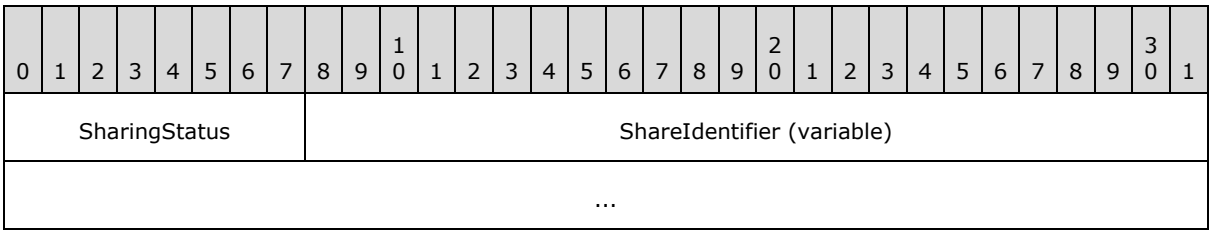


ExecuteCommand (variable): An array of ANSI characters that specifies the **ExecuteCommandType**. Note that this array is not zero-terminated. The ExecuteCommand value MUST be one of the **ExecuteCommandType** constants (section 2.2.1.3).

ShareName (variable): An optional, null-terminated ANSI string that specifies the share on which the command operates. This field MUST NOT be present if **ExecuteCommand** is CMD_INITSHARE, and it MUST be present if **ExecuteCommand** is not CMD_INITSHARE.

2.2.2.2 SHARE_LIST_ENTRYA

The SHARE_LIST_ENTRYA structure contains information about a shared clipbook.



SharingStatus (1 byte): An 8-bit unsigned integer that specifies the **SharingStatusType (section 2.2.1.6)**.

ShareIdentifier (variable): An array of ANSI characters that specifies the name of the clipbook share. Note: this array is NOT zero-terminated.

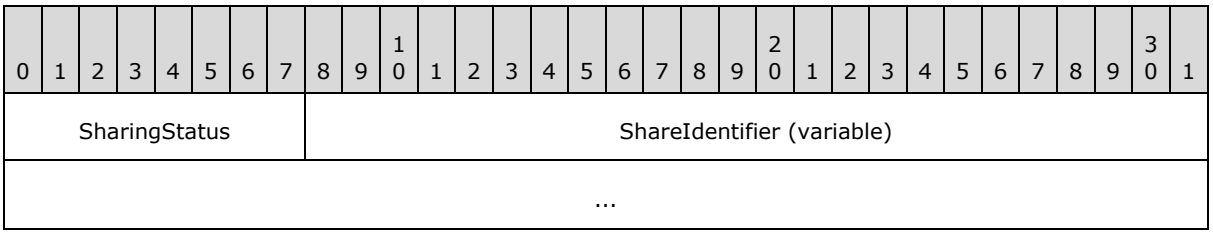
2.2.2.3 SHARE_LISTA

The SHARE_LISTA structure contains information about a set of clipbooks. A SHARE_LISTA structure conforms to the following ABNF [RFC5234]:

```
SHARE_LISTA = SHARE_LIST_ENTRYA *(%x09 SHARE_LIST_ENTRYA) %x00
```

2.2.2.4 SHARE_LIST_ENTRYW

The SHARE_LIST_ENTRYW structure contains information about a shared clipbook.



SharingStatus (1 byte): A 16-bit unsigned integer that specifies the **SharingStatusType (section 2.2.1.6)**.

ShareIdentifier (variable): An array of Unicode characters that specifies the name of the clipbook share. Note: this array is NOT zero-terminated.

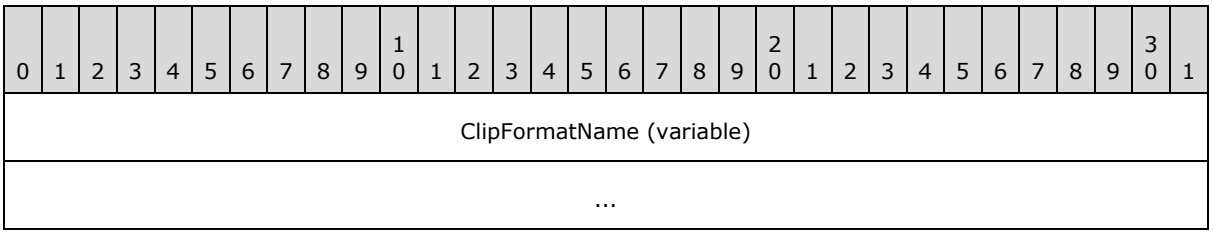
2.2.2.5 SHARE_LISTW

The SHARE_LISTW structure contains information about a set of clipbooks. A SHARE_LISTW structure conforms to the following ABNF [RFC5234]:

```
SHARE_LISTW = SHARE_LIST_ENTRYW *(%x00.09 SHARE_LIST_ENTRYW) %x00.00
```

2.2.2.6 CLIPFORMAT_LIST_ENTRYA

The CLIPFORMAT_LIST_ENTRYA structure describes a clipboard format.



ClipFormatName (variable): An array of ANSI characters that specifies the name of the clipboard format. For predefined clipboard formats, this value is a ClipboardFormatName constant (section 2.2.1.1). Note: this array is NOT zero-terminated.<1>

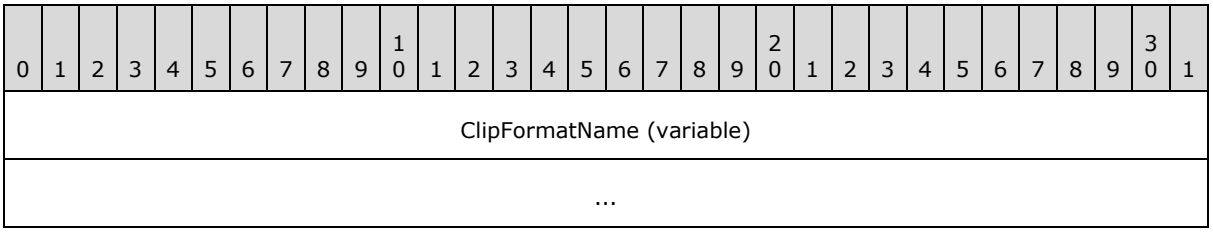
2.2.2.7 CLIPFORMAT_LISTA

The CLIPFORMAT_LISTA structure describes a set of clipboard formats. A CLIPFORMAT_LISTA structure conforms to the following ABNF [RFC5234]:

```
CLIPFORMAT_LISTA = CLIPFORMAT_LIST_ENTRYA *(%x09 CLIPFORMAT_LIST_ENTRYA) %x00
```

2.2.2.8 CLIPFORMAT_LIST_ENTRYW

The CLIPFORMAT_LIST_ENTRYW structure describes a clipboard format.



ClipFormatName (variable): An array of Unicode characters that specifies the name of the clipboard format. For predefined clipboard formats, this value is a **ClipboardFormatName (section 2.2.1.1)** constant. Note this array is NOT zero-terminated.<2>

2.2.2.9 CLIPFORMAT_LISTW

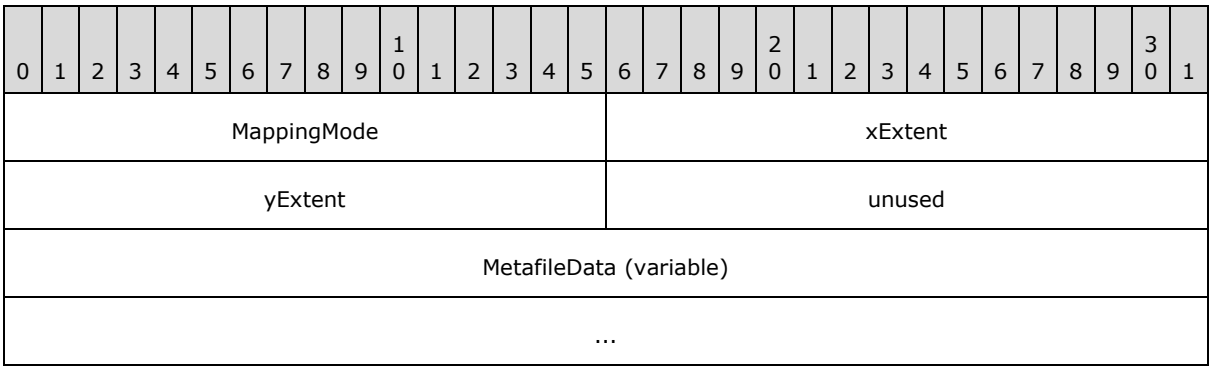
The CLIPFORMAT_LISTW structure describes a set of clipboard formats. A CLIPFORMAT_LISTW structure conforms to the following ABNF [RFC5234]:

```
CLIPFORMAT_LISTW = CLIPFORMAT_LIST_ENTRYW *(%x00.09 CLIPFORMAT_LIST_ENTRYW) %x00.00
```

2.2.3 Clipbook Data

2.2.3.1 CLIPDATA_METAFILEPICT

The CLIPDATA_METAFILEPICT structure contains metafile data.<3>



MappingMode (2 bytes): A 16-bit unsigned integer that specifies the **mapping mode** in which this picture is to be drawn.

xExtent (2 bytes): A 16-bit unsigned integer that specifies the width of the rectangle within which the picture is to be drawn. The coordinates are in units that correspond to the mapping mode.

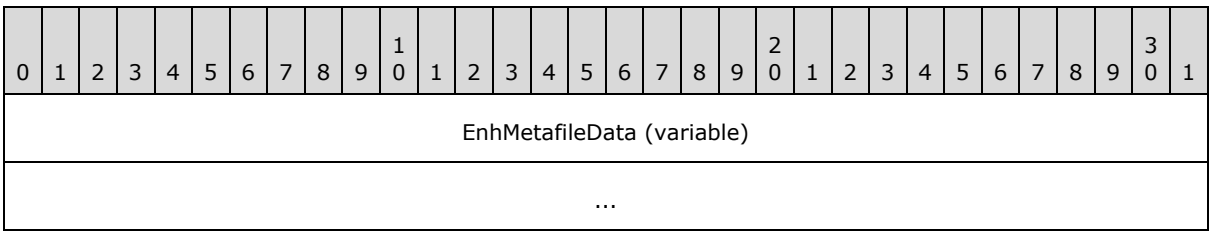
yExtent (2 bytes): A 16-bit unsigned integer that specifies the height of the rectangle within which the picture is to be drawn. The coordinates are in units that correspond to the mapping mode.

unused (2 bytes): Unused. SHOULD be zero.

MetafileData (variable): Data corresponding to a memory-based **Windows metafile format (WMF)** metafile [MS-WMF].<4>

2.2.3.2 CLIPDATA_ENHMETAFILE

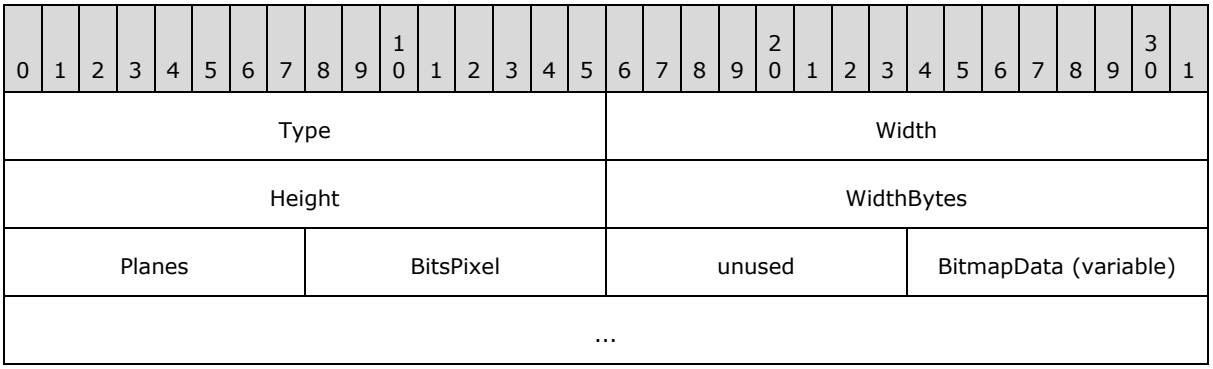
The CLIPDATA_ENHMETAFILE structure contains EMF data.



EnhMetafileData (variable): Data corresponding to a memory-based EMF metafile [MS-EMF].<5>

2.2.3.3 CLIPDATA_BITMAP

The CLIPDATA_BITMAP structure contains bitmap data.



Type (2 bytes): A 16-bit unsigned integer that specifies the bitmap type. MUST be set to 0x0000.

Width (2 bytes): A 16-bit unsigned integer that specifies the width, in pixels, of the bitmap.

Height (2 bytes): A 16-bit unsigned integer that specifies the height, in pixels, of the bitmap.

WidthBytes (2 bytes): A 16-bit unsigned integer that specifies the number of bytes in each scan line. This value MUST be divisible by 2, because the system assumes that the bit values in a bitmap form an array that is word aligned.

Planes (1 byte): An 8-bit unsigned integer that specifies the count of **color planes**.

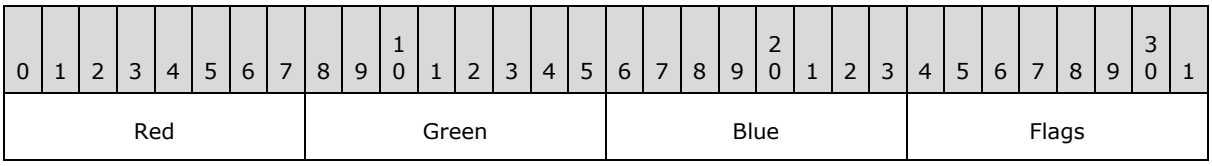
BitsPixel (1 byte): An 8-bit unsigned integer that specifies the number of bits required to indicate the color of a pixel.

unused (1 byte): Unused. SHOULD be zero.

BitmapData (variable): An array of byte values forming the bitmap data as specified by the previous fields.

2.2.3.4 CLIPDATA_PALETTE_ENTRY

The CLIPDATA_PALETTE_ENTRY structure contains palette color information.



Red (1 byte): An 8-bit unsigned integer that specifies the red **intensity** value for the palette entry.

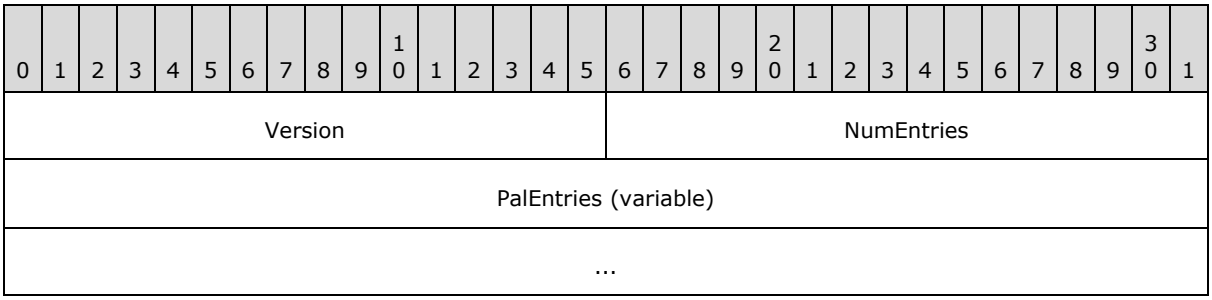
Green (1 byte): An 8-bit unsigned integer that specifies the green intensity value for the palette entry.

Blue (1 byte): An 8-bit unsigned integer that specifies the blue intensity value for the palette entry.

Flags (1 byte): An 8-bit unsigned integer that specifies the **PaletteEntryFlags (section 2.2.1.5)** usage of the palette entry.

2.2.3.5 CLIPDATA_PALETTE

The CLIPDATA_PALETTE structure contains palette data.



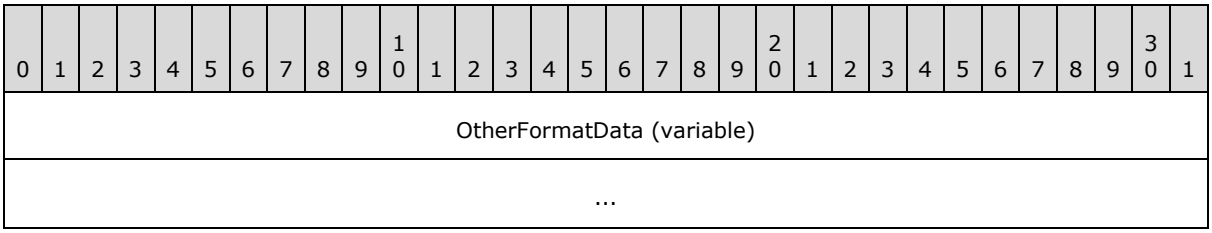
Version (2 bytes): A 16-bit signed integer that specifies the version number of the system. MUST be set to 0x0300.

NumEntries (2 bytes): A 16-bit unsigned integer that specifies the number of entries in **PalEntries**.

PalEntries (variable): A series of **NumEntries** CLIPDATA_PALETTE_ENTRY (section 2.2.3.4) structures that specifies the palette information.

2.2.3.6 CLIPDATA_OTHERFORMATS

The CLIPDATA_OTHERFORMATS structure contains data corresponding to arbitrary clipboard formats.



OtherFormatData (variable): Data corresponding to arbitrary clipboard formats.<6>

3 Protocol Details

In the Desktop Clipboard Protocol, a server allows a client to access a set of shared clipbooks. Implementations can simultaneously participate in both client and server roles to allow for two-way sharing of clipbooks. Implementations can represent the local clipboard as a clipbook shared by a server, so that client software can interact agnostically with the local clipboard and remote clipbooks.

In the following diagram, a client and server first negotiate a DDE conversation as described in sections 3.1.3 and 3.2.3.

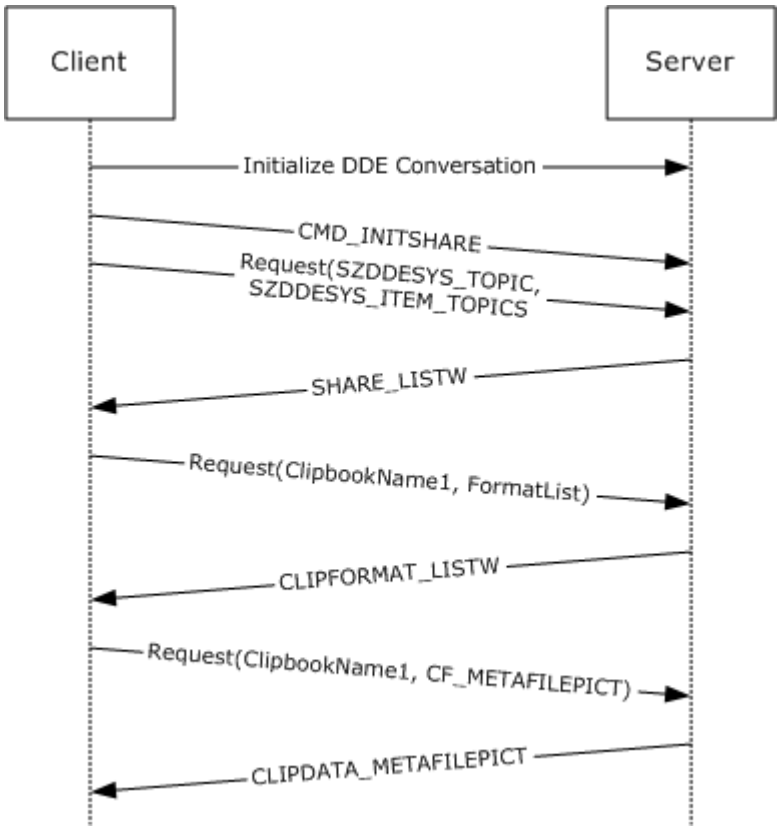


Figure 2: Clipboard sharing session

The following messages are then transmitted:

1. The client sends a CMD_INITSHARE (section 3.1.5.1.1) message, directing the server to initialize its **SharedClipbookData** information.
2. The client sends a DDE request with SZDDESYS_TOPIC as the topic and SZDDESYS_ITEM_TOPICS as the item (section 2.2.1.2).
3. The server sends a SHARE_LISTW (section 2.2.2.5) message containing the set of shared clipbooks.
4. Knowing the available clipbooks, the client sends a DDE request with ClipbookName1 as the topic and "FormatList" as the item.

5. The server sends a CLIPFORMAT_LISTW (section 2.2.2.9) message containing the set of supported formats for the ClipbookName1 clipbook.

6. Knowing the supported clipboard formats, the client sends a DDE request with ClipbookName1 as the topic and **CF_METAFILEPICT** as the item.

7. The server sends a CLIPDATA_METAFILEPICT (section 2.2.3.1) message containing the metafile information for the ClipbookName1 clipbook. The client renders this information to the user.

In this exchange, the set of supported formats returned in step 5 includes CF_METAFILEPICT; but in step 6, the client can request any other clipboard format, using one of the **ClipboardFormatName (section 2.2.1.1)** values that is supported in the set returned in step 5. The client can continue by requesting other formats.

3.1 Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that a server implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

Abstractly, clipbook information is kept as follows:

ClipbookData.ClipbookName: A string uniquely identifying the clipbook.

ClipbookData.SharingStatus: The **SharingStatusType (section 2.2.1.6)** value of the clipbook.

ClipbookData.Formats: An array of clipboard formats present in the clipbook.

ClipbookData.Formats[N].FormatName: The **ClipboardFormatName (section 2.2.1.1)** value identifying the clipboard format within the clipbook.

ClipbookData.Formats[N].Data: The data contained for the clipboard format within the clipbook.

Servers of the Desktop Clipboard Protocol SHOULD maintain the following state.

SharedClipbookData: An array of **ClipbookData** entries representing the clipbooks shared by the server.

3.1.2 Timers

None.

3.1.3 Initialization

A NetDDE server is initialized as follows. See [MSDN-NETDDE] for additional information.

The NetDDE server machine creates a static network DDE share using NDdeShareAdd. The DDE share MUST be named "CLPBK$" and MUST have the static topic list of "ClipSrv|System".

The client initiates the NetDDE conversation as specified in section 3.2.3.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

Malformed, unrecognized, and out-of-sequence packets MUST be ignored by the server.

3.1.5.1 Command Message Processing

The server MUST handle command messages received as XTYP_EXECUTE notifications (section 2.2.1.4) to its **DdeCallback** function. An EXECCOMMAND (section 2.2.2.1) data block is extracted using the **DdeGetData** function and is processed as follows.

3.1.5.1.1 CMD_INITSHARE

The server MUST perform any initialization required to populate **SharedClipbookData**. An implementation can load shared clipbook data from a user-configured persisted storage location.

3.1.5.1.2 CMD_DELETE

The server MUST remove the **ClipbookData** with the **ClipbookName** specified by **EXECCOMMAND.ShareName** from **SharedClipbookData**.

3.1.5.1.3 CMD_SHARE

The server MUST find the **ClipbookData** with the **ClipbookName** specified by **EXECCOMMAND.ShareName** and set **SharingStatus** to STATUS_SHARED.

3.1.5.1.4 CMD_UNSHARE

The server MUST find the **ClipbookData** with the **ClipbookName** specified by **EXECCOMMAND.ShareName** and set **SharingStatus** to STATUS_UNSHARED.

3.1.5.1.5 CMD_PASTE

The server MUST add the **ClipbookData** with the **ClipbookName** specified by **EXECCOMMAND.ShareName** to **SharedClipbookData**.

3.1.5.2 Responses to Data Requests

The server MUST handle data request messages received as XTYP_ADVREQ or **XTYP_REQUEST** notifications (section 2.2.1.4) to its **DdeCallback** function.

If the **DdeCallback** DDE topic is SZDDESYS_TOPIC and the **DdeCallback** item is SZDDESYS_ITEM_TOPICS (section 2.2.1.2):

- If the requested clipboard format is CF_TEXT, the server MUST respond by creating a SHARE_LISTA (section 2.2.2.3) structure corresponding with **SharedClipbookData** and returning it via DdeCreateDataHandle.
- If the requested clipboard format is CF_UNICODETEXT, the server MUST respond by creating a SHARE_LISTW (section 2.2.2.5) structure corresponding with **SharedClipbookData** and returning it via DdeCreateDataHandle.

If the **DdeCallback** item is "FormatList":

- If the requested clipboard format is CF_TEXT, the server MUST respond by creating a CLIPFORMAT_LISTA (section 2.2.2.7) structure corresponding to the **ClipbookData** with the **ClipbookName** specified by the **DdeCallback** topic and returning it via DdeCreateDataHandle.
- If the requested clipboard format is CF_UNICODETEXT, the server MUST respond by creating a CLIPFORMAT_LISTW (section 2.2.2.9) structure corresponding to the **ClipbookData** with the **ClipbookName** specified by the **DdeCallback** topic and returning it via DdeCreateDataHandle.

In all other cases, the server MUST find the clipboard data with the **ClipbookName** specified by the **DdeCallback** topic and the **FormatName** specified by the **DdeCallback** item. This data is then serialized and returned to the client via DdeCreateDataHandle as follows:

- If the requested clipboard format is CF_ENHMETAFILE, the server MUST return a CLIPDATA_ENHMETAFILE (section 2.2.3.2) structure.
- If the requested clipboard format is CF_METAFILEPICT, the server MUST return a CLIPDATA_METAFILEPICT (section 2.2.3.1) structure.
- If the requested clipboard format is CF_PALETTE, the server MUST return a CLIPDATA_PALETTE (section 2.2.3.5) structure.
- If the requested clipboard format is CF_BITMAP or CF_DIB, the server MUST return a CLIPDATA_BITMAP (section 2.2.3.3) structure.
- Otherwise, the server MUST return a CLIPDATA_OTHERFORMATS (section 2.2.3.6) structure.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible data organization that a client implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

The Desktop Clipboard Protocol provides the means for clients to request the **SharedClipbookData** information from the server, including enumerating the shared clipbooks and retrieving the clipbook data.

A client application can render this information in a form visible to the user, allow the user to copy the clipbook contents into their local system clipboard, or allow the user to save the clipbook contents.

3.2.2 Timers

None.

3.2.3 Initialization

A NetDDE conversation is initialized by a client as follows. See [MSDN-NETDDE] for additional information.

1. The NetDDE client machine initiates the DDE conversation using DdeConnect. The DdeConnect service name MUST be of the form "\\computername\NDDE$", where computername is the **NetBIOS** name of the server machine. The DdeConnect topic name MUST be "CLPBK$".

2. The client then sends an XTYP_EXECUTE notification (section 2.2.1.4) to the server with a CMD_INITSHARE (section 3.1.5.1.1) message as the contents of DdeClientTransaction.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

Malformed, unrecognized, and out-of-sequence packets MUST be ignored by the client.

3.2.5.1 Command Messages

A client can send command messages as XTYP_EXECUTE notifications (section 2.2.1.4) to the server's **DdeCallback** function. An EXECCOMMAND (section 2.2.2.1) data block is set up according to the specific command.

3.2.5.2 Data Requests

A client can send data request messages as XTYP_ADVREQ or XTYP_REQUEST notifications (section 2.2.1.4) to the server's **DdeCallback** function.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

In this example, the server machine is sharing a clipboard entry called "ShareName" which contains "Sample Text" as its textual data.

1. The client makes a SZDDESYS_ITEM_TOPICS request (section 2.2.1.2) of the server.

   The following is the hexadecimal representation of the SHARE_LISTA (section 2.2.2.3) data the server returns:

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0000	3F	09	24	53	68	61	72	65	4E	61	6D	65	00	00	00	00

   - SharingStatus: (1 byte, offset 0x0000), 0x3F is STATUS_UPDATED.
   - ShareIdentifier: (0 bytes, offset 0x0001), empty value.
   - Tab delimiter: (1 byte, offset 0x0001), 0x09 as required.
   - SharingStatus: (1 byte, offset 0x0002), 0x24 is STATUS_SHARED.
   - ShareIdentifier: (9 bytes, offset 0x0003), "ShareName" in **ASCII**.
   - Null terminator: (1 byte, offset 0x000C), 0x00 as required.

2. The client recognizes the "ShareName" share is shared and requests its supported clipboard formats with the "FormatList" DDE message.

   The following is the hexadecimal representation of the CLIPFORMAT_LISTA (section 2.2.2.7) data the server returns:

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0000	26	55	6E	69	63	6F	64	65	20	54	65	78	74	09	09	26
0010	54	65	78	74	09	26	4F	45	4D	20	54	65	78	74	09	43
0000	6C	69	70	62	6F	6F	6B	20	50	72	65	76	69	65	77	00

   - ClipFormatName: (13 bytes, offset 0x0000), "&Unicode Text" in ASCII.
   - Tab delimiter: (1 byte, offset 0x000D), 0x09 as required.
   - ClipFormatName: (0 bytes, offset 0x000E), empty value.
   - Tab delimiter: (1 byte, offset 0x000E), 0x09 as required.
   - ClipFormatName: (5 bytes, offset 0x000F), "&Text" in ASCII.
   - Tab delimiter: (1 byte, offset 0x0014), 0x09 as required.
   - ClipFormatName: (9 bytes, offset 0x0015), "&OEM Text" in ASCII.
   - Tab delimiter: (1 byte, offset 0x001E), 0x09 as required.
   - ClipFormatName: (16 bytes, offset 0x001F), "Clipbook Preview" in ASCII.
   - Null terminator: (1 byte, offset 0x002F), 0x00 as required.

3. The client determines that Unicode text is its preferred format for consumption of the data, so it makes a request for CF_UNICODETEXT.

   The following is the hexadecimal representation of the CLIPDATA_OTHERFORMATS (section 2.2.3.6) data the server returns:

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0000	53	00	61	00	6D	00	70	00	6C	00	65	00	20	00	54	00
0010	65	00	78	00	74	00	00	00								

   - OtherFormatData: (20 bytes, offset 0x0000), "Sample Text" in Unicode characters.
   - Null terminator: (2 bytes, offset 0x0016), 0x0000 as required.

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft Windows® 2000 operating system Service Pack 4 (SP4)
- Windows® XP operating system
- Windows Server® 2003 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 2.2.2.6: In Windows environments, for non-predefined clipboard formats the clipboard name corresponds with return values of the **GetClipboardFormatName** function [MSDN-CLIPBOARD].

<2> Section 2.2.2.8: In Windows environments, for non-predefined clipboard formats, the clipboard name corresponds with return values of the **GetClipboardFormatName** function [MSDN-CLIPBOARD].

<3> Section 2.2.3.1: In Windows environments, these structure fields map to the **METAFILEPICT** structure [MSDN-METAFILEPICT].

<4> Section 2.2.3.1: In Windows environments, this is the data used by the **SetMetaFileBitsEx** function and returned by the **GetMetaFileBitsEx** function [MSDN-META].

<5> Section 2.2.3.2: In Windows environments, this is the data used by the **SetEnhMetaFileBits** function and returned by the **GetEnhMetaFileBits** function [MSDN-META].

<6> Section 2.2.3.6: In Windows environments, the arbitrary clipboard format data is obtained by performing a GlobalLock operation on the handle returned by the **GetClipboardData** function [MSDN-CLIPBOARD].

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index