

# [MS-EMF]: Enhanced Metafile Format Specification

---

## Intellectual Property Rights Notice for Protocol Documentation

- This protocol 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 protocols, and may distribute portions of it in your implementations of the protocols or your documentation as necessary to properly document the implementation. This permission also applies to any documents that are referenced in the protocol documentation.
- Microsoft does not claim any trade secret rights in this documentation.
- Microsoft has patents that may cover your implementations of the protocols. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. If you are interested in obtaining a patent license, please contact [protocol@microsoft.com](mailto:protocol@microsoft.com).
- 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.
- All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

This protocol documentation is intended for use in conjunction with publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

A protocol specification does 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.

## Revision Summary

Date	Revision History	Revision Class	Comments
04/03/2007	0.01		MCPP Milestone Longhorn Initial Availability
07/03/2007	1.0	Major	MLonghorn+90
07/20/2007	2.0	Major	Restructured record sections according to category; other updates.
08/10/2007	2.1	Minor	Updated the technical content.

<b>Date</b>	<b>Revision History</b>	<b>Revision Class</b>	<b>Comments</b>
09/28/2007	2.2	Minor	Updated the technical content.
10/23/2007	3.0	Major	Added new sections describing the EMR_COMMENT_EMFPLUS and EMR_COMMENT_EMFSPOOL records.
11/30/2007	3.1	Minor	Standardized art.
01/25/2008	3.2	Minor	Reconstructed record categories for clarity.

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>8</b>
1.1	Glossary .....	8
1.2	References .....	13
1.2.1	Normative References .....	13
1.2.2	Informative References.....	14
1.3	Structure Overview (Synopsis) .....	14
1.3.1	Metafile Structure .....	14
1.3.2	Graphics Objects .....	16
1.3.3	Byte Ordering .....	17
1.4	Relationship to Protocols and Other Structures .....	17
1.5	Applicability Statement .....	17
1.6	Versioning and Localization .....	17
1.7	Vendor-Extensible Fields .....	18
<b>2</b>	<b>Structures.....</b>	<b>19</b>
2.1	EMF Enumerations.....	19
2.1.1	RecordType Enumeration .....	19
2.1.2	ArcDirection Enumeration .....	28
2.1.3	ArmStyle Enumeration.....	28
2.1.4	BackgroundMode Enumeration.....	29
2.1.5	ColorAdjustment Enumeration .....	29
2.1.6	ColorMatchToTarget Enumeration.....	29
2.1.7	ColorSpace Enumeration .....	30
2.1.8	Contrast Enumeration.....	30
2.1.9	DIBColors Enumeration.....	31
2.1.10	EmrComment Enumeration .....	31
2.1.11	ExtTextOutOptions Enumeration .....	32
2.1.12	FamilyType Enumeration.....	33
2.1.13	FloodFill Enumeration .....	33
2.1.14	FormatSignature Enumeration .....	34
2.1.15	GradientFill Enumeration.....	34
2.1.16	GraphicsMode Enumeration .....	35
2.1.17	HatchStyle Enumeration .....	35
2.1.18	ICMMode Enumeration.....	36
2.1.19	Illuminant Enumeration .....	36
2.1.20	Letterform Enumeration.....	37
2.1.21	MapMode Enumeration .....	38
2.1.22	MetafileVersion Enumeration .....	39
2.1.23	MidLine Enumeration.....	39
2.1.24	ModifyWorldTransformMode Enumeration .....	40
2.1.25	PenStyle Enumeration .....	41
2.1.26	Point Enumeration .....	42
2.1.27	PolygonFillMode Enumeration .....	42
2.1.28	Proportion Enumeration .....	43
2.1.29	RegionMode Enumeration.....	43
2.1.30	SerifType Enumeration .....	44
2.1.31	StockObject Enumeration.....	45
2.1.32	StretchMode Enumeration .....	46
2.1.33	StrokeVariation Enumeration .....	47
2.1.34	TextAlignmentMode Enumeration .....	48
2.1.35	VerticalTextAlignmentMode Enumeration.....	48
2.1.36	Weight Enumeration.....	49

2.1.37	XHeight Enumeration .....	49
2.2	EMF Objects .....	50
2.2.1	ColorAdjustment Object.....	50
2.2.2	DesignVector Object.....	51
2.2.3	EmrFormat Object .....	52
2.2.4	EmrText Object .....	53
2.2.5	EpsData Object .....	55
2.2.6	GradientRectangle Object.....	55
2.2.7	GradientTriangle Object.....	56
2.2.8	HeaderExtension1 Object.....	56
2.2.9	HeaderExtension2 Object.....	57
2.2.10	LogBrushEx Object.....	57
2.2.11	LogFont Object.....	58
2.2.12	LogFontEx Object .....	61
2.2.13	LogFontExDv Object.....	63
2.2.14	LogFontPanose Object .....	64
2.2.15	LogPalette Object .....	67
2.2.16	LogPaletteEntry Object .....	67
2.2.17	LogPen Object.....	68
2.2.18	LogPenEx Object.....	68
2.2.19	Panose Object .....	70
2.2.20	PixelFormatDescriptor Object.....	71
2.2.21	RegionData Object .....	75
2.2.22	RegionDataHeader Object .....	75
2.2.23	TriVertex Object .....	76
2.2.24	UniversalFontId Object .....	77
2.2.25	XForm Object.....	78
2.3	EMF Records.....	80
2.3.1	Bitmap Record Types.....	80
2.3.1.1	EMR_ALPHABLEND Record .....	82
2.3.1.2	EMR_BITBLT Record .....	87
2.3.1.3	EMR_MASKBLT Record .....	89
2.3.1.4	EMR_PLGBLT Record.....	93
2.3.1.5	EMR_SETDIBITSTODEVICE Record.....	97
2.3.1.6	EMR_STRETCHBLT Record .....	100
2.3.1.7	EMR_STRETCHDIBITS Record .....	103
2.3.1.8	EMR_TRANSPARENTBLT Record.....	105
2.3.2	Clipping Record Types .....	108
2.3.2.1	EMR_EXCLUDECLIPRECT Record .....	109
2.3.2.2	EMR_EXTSELECTCLIPRGN Record .....	110
2.3.2.3	EMR_INTERSECTCLIPRECT Record .....	111
2.3.2.4	EMR_OFFSETCLIPRGN Record .....	112
2.3.2.5	EMR_SELECTCLIPPATH Record .....	113
2.3.3	Comment Record Types.....	113
2.3.3.1	EMR_COMMENT Record .....	114
2.3.3.2	EMR_COMMENT_EMFPLUS Record.....	115
2.3.3.3	EMR_COMMENT_EMFSPOOL Record .....	116
2.3.3.4	EMR_COMMENT_PUBLIC Record Types .....	117
2.3.3.4.1	EMR_COMMENT_BEGINGROUP Record .....	118
2.3.3.4.2	EMR_COMMENT_ENDGROUP Record .....	120
2.3.3.4.3	EMR_COMMENT_MULTIFORMATS Record.....	120
2.3.3.4.4	EMR_COMMENT_WINDOWS_METAFILE Record.....	122
2.3.4	Control Record Types .....	123
2.3.4.1	EMR_EOF Record.....	124
2.3.4.2	EMR_HEADER Record Types.....	125

2.3.4.2.1	EmfMetafileHeader Record .....	128
2.3.4.2.2	EmfMetafileHeaderExtension1 Record .....	129
2.3.4.2.3	EmfMetafileHeaderExtension2 Record .....	130
2.3.5	Drawing Record Types .....	132
2.3.5.1	EMR_ANGLEARC Record .....	136
2.3.5.2	EMR_ARC Record .....	137
2.3.5.3	EMR_ARCTO Record .....	138
2.3.5.4	EMR_CHORD Record .....	139
2.3.5.5	EMR_ELLIPSE Record .....	141
2.3.5.6	EMR_EXTFLOODFILL Record .....	141
2.3.5.7	EMR_EXTTEXTOUTA Record .....	142
2.3.5.8	EMR_EXTTEXTOUTW Record .....	144
2.3.5.9	EMR_FILLPATH Record .....	145
2.3.5.10	EMR_FILLRGN Record .....	145
2.3.5.11	EMR_FRAMERGN Record .....	146
2.3.5.12	EMR_GRADIENTFILL Record .....	148
2.3.5.13	EMR_LINETO Record .....	150
2.3.5.14	EMR_PAINTRGN Record .....	150
2.3.5.15	EMR_PIE Record .....	151
2.3.5.16	EMR_POLYBEZIER Record .....	152
2.3.5.17	EMR_POLYBEZIER16 Record .....	154
2.3.5.18	EMR_POLYBEZIERTO Record .....	155
2.3.5.19	EMR_POLYBEZIERTO16 Record .....	156
2.3.5.20	EMR_POLYDRAW Record .....	157
2.3.5.21	EMR_POLYDRAW16 Record .....	158
2.3.5.22	EMR_POLYGON Record .....	159
2.3.5.23	EMR_POLYGON16 Record .....	160
2.3.5.24	EMR_POLYLINE Record .....	161
2.3.5.25	EMR_POLYLINE16 Record .....	162
2.3.5.26	EMR_POLYLINETO Record .....	163
2.3.5.27	EMR_POLYLINETO16 Record .....	164
2.3.5.28	EMR_POLYPOLYGON Record .....	165
2.3.5.29	EMR_POLYPOLYGON16 Record .....	166
2.3.5.30	EMR_POLYPOLYLINE Record .....	168
2.3.5.31	EMR_POLYPOLYLINE16 Record .....	169
2.3.5.32	EMR_POLYTEXTOUTA Record .....	171
2.3.5.33	EMR_POLYTEXTOUTW Record .....	172
2.3.5.34	EMR_RECTANGLE Record .....	173
2.3.5.35	EMR_ROUNDRECT Record .....	174
2.3.5.36	EMR_SETPIXELV Record .....	174
2.3.5.37	EMR_SMALLTEXTOUT Record .....	175
2.3.5.38	EMR_STROKEANDFILLPATH Record .....	177
2.3.5.39	EMR_STROKEPATH Record .....	178
2.3.6	Escape Record Types .....	178
2.3.6.1	EMR_DRAWESCAPE Record .....	179
2.3.6.2	EMR_EXTESCAPE Record .....	180
2.3.6.3	EMR_NAMEDESCAPE Record .....	181
2.3.7	Object Creation Record Types .....	182
2.3.7.1	EMR_CREATEBRUSHINDIRECT Record .....	184
2.3.7.2	EMR_CREATECOLORSPACE Record .....	184
2.3.7.3	EMR_CREATECOLORSPACEW Record .....	185
2.3.7.4	EMR_CREATEDIBPATTERNBRUSHPT Record .....	186
2.3.7.5	EMR_CREATEMONOBRUSH Record .....	187
2.3.7.6	EMR_CREATEPALETTE Record .....	188
2.3.7.7	EMR_CREATEPEN Record .....	189

2.3.7.8	EMR_EXTCREATEFONTINDIRECTW Record .....	190
2.3.7.9	EMR_EXTCREATEPEN Record .....	192
2.3.8	Object Manipulation Record Types .....	193
2.3.8.1	EMR_COLORCORRECTPALETTE Record .....	194
2.3.8.2	EMR_DELETECOLORSPACE Record .....	195
2.3.8.3	EMR_DELETEOBJECT Record .....	196
2.3.8.4	EMR_RESIZEPALETTE Record .....	197
2.3.8.5	EMR_SELECTOBJECT Record .....	197
2.3.8.6	EMR_SELECTPALETTE Record .....	198
2.3.8.7	EMR_SETCOLORSPACE Record .....	199
2.3.8.8	EMR_SETPALETTEENTRIES Record .....	199
2.3.9	OpenGL Record Types .....	200
2.3.9.1	EMR_GLSBOUNDEDRECORD Record .....	201
2.3.9.2	EMR_GLSRECORD Record .....	202
2.3.10	Path Bracket Record Types .....	203
2.3.11	State Record Types .....	204
2.3.11.1	EMR_COLORMATCHTOTARGETW Record .....	208
2.3.11.2	EMR_FORCEUFIMAPPING Record .....	209
2.3.11.3	EMR_INVERTGRN Record .....	210
2.3.11.4	EMR_MOVETOEX Record .....	210
2.3.11.5	EMR_PIXELFORMAT Record .....	211
2.3.11.6	EMR_RESTOREDC Record .....	212
2.3.11.7	EMR_SCALEVIEWPORTEXTEX Record .....	213
2.3.11.8	EMR_SCALEWINDOWEXTEX Record .....	214
2.3.11.9	EMR_SETARCDIRECTION Record .....	215
2.3.11.10	EMR_SETBKCOLOR Record .....	216
2.3.11.11	EMR_SETBKMODE Record .....	216
2.3.11.12	EMR_SETBRUSHORGEX Record .....	217
2.3.11.13	EMR_SETCOLORADJUSTMENT Record .....	217
2.3.11.14	EMR_SETICMMODE Record .....	218
2.3.11.15	EMR_SETICMPROFILEA Record .....	219
2.3.11.16	EMR_SETICMPROFILEW Record .....	220
2.3.11.17	EMR_SETLAYOUT Record .....	221
2.3.11.18	EMR_SETLINKEDUFIS Record .....	222
2.3.11.19	EMR_SETMAPMODE Record .....	223
2.3.11.20	EMR_SETMAPPERFLAGS Record .....	224
2.3.11.21	EMR_SETMITERLIMIT Record .....	225
2.3.11.22	EMR_SETPOLYFILLMODE Record .....	225
2.3.11.23	EMR_SETROP2 Record .....	226
2.3.11.24	EMR_SETSTRETCHBLTMODE Record .....	226
2.3.11.25	EMR_SETTEXTALIGN Record .....	227
2.3.11.26	EMR_SETTEXTCOLOR Record .....	228
2.3.11.27	EMR_SETTEXTJUSTIFICATION Record .....	228
2.3.11.28	EMR_SETVIEWPORTEXTEX Record .....	229
2.3.11.29	EMR_SETVIEWPORTORGEX Record .....	230
2.3.11.30	EMR_SETWINDOWEXTEX Record .....	230
2.3.11.31	EMR_SETWINDOWORGEX Record .....	231
2.3.12	Transform Record Types .....	231
2.3.12.1	EMR_MODIFYWORLDTRANSFORM Record .....	232
2.3.12.2	EMR_SETWORLDTRANSFORM Record .....	233
<b>3</b>	<b>Structure Examples .....</b>	<b>235</b>
3.1	Metafile Design .....	235
3.1.1	Managing Objects .....	235
3.1.1.1	EMF Object Table .....	235

3.1.2	Byte Ordering .....	236
3.1.3	Run-Length Encoding (RLE) Bitmap Compression .....	236
3.2	EMF Metafile Example .....	238
3.2.1	EMR_HEADER Example .....	252
3.2.2	EMR_CREATEBRUSHINDIRECT Example .....	255
3.2.3	EMR_SELECTOBJECT Example .....	256
3.2.4	EMR_BITBLT Example.....	257
3.2.5	EMR_SELECTOBJECT Example .....	259
3.2.6	EMR_BITBLT Example.....	260
3.2.7	EMR_SETBKMODE Example .....	275
3.2.8	EMR_EXTCREATEFONTINDIRECTW Example .....	275
3.2.9	EMR_SELECTOBJECT Example .....	278
3.2.10	EMR_EXTTEXTOUTW Example.....	279
3.2.11	EMR_EXTCREATEFONTINDIRECTW Example .....	280
3.2.12	EMR_SELECTOBJECT Example .....	283
3.2.13	EMR_EXTCREATEFONTINDIRECTW Example .....	284
3.2.14	EMR_SELECTOBJECT Example .....	287
3.2.15	EMR_DELETEOBJECT Example .....	287
3.2.16	EMR_EXTCREATEFONTINDIRECTW Example .....	288
3.2.17	EMR_SELECTOBJECT Example .....	290
3.2.18	EMR_SELECTOBJECT Example .....	291
3.2.19	EMR_DELETEOBJECT Example .....	291
3.2.20	EMR_DELETEOBJECT Example .....	292
3.2.21	EMR_SELECTOBJECT Example .....	292
3.2.22	EMR_EOF Example .....	293
<b>4</b>	<b>Security Considerations .....</b>	<b>294</b>
<b>5</b>	<b>Appendix A: Windows Behavior .....</b>	<b>295</b>
<b>6</b>	<b>Index.....</b>	<b>299</b>

# 1 Introduction

The Enhanced Metafile Format (EMF) is a collection of records that can store a picture in a device-independent format. The stored picture can be rendered by parsing and processing the **EMF** structure.

An EMF **metafile** is a series of variable-length records, called metafile records, which contain EMF structures. The metafile—also called a **vector image**—begins with a header record, which may include the version of the metafile, its size, the resolution of the device on which the picture was created, and the dimensions of the picture. A metafile is "played back" when its records are converted to a format understood by a specific graphics device. The image defined in an EMF structure maintains its dimensions, shape, and proportions on any output device, including printers, plotters, desktops, or in the client areas of many applications.

## 1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

**ANSI Character Set**  
**ASCII**  
**Big-Endian**  
**Color Profile**  
**Little-Endian**  
**Original Equipment Manufacturer (OEM) character set**  
**PostScript**  
**Print Job**  
**Print Server**  
**Printer Driver**  
**Spool File**  
**Unicode**  
**UTF16-LE**

The following terms are specific to this document:

**Additive Color Model:** A **color model**, which involves light emitted directly from a source or illuminant of some sort. The additive reproduction process usually uses red, green and blue light to produce the other colors.

**Alpha Transparency:** An alpha value is a transparency value represented by a number between zero and one. Each pixel has an alpha value that represents its level of transparency, which is multiplied by the color values to get the final value.

**Bezier Curve:** A type of curve defined by a mathematical formula and a number of points greater than or equal to 2, used in computer graphics and in the mathematical field of numeric analysis. A cubic Bezier curve is defined by four points: two endpoints and two control points. The curve does not pass through the control points, but the control points act like magnets, pulling the curve in certain directions and influencing the way the curve bends. With multiple Bezier curves, the endpoint of one is the starting point of the next.

**Bitmap:** A collection of structures that contain a device-independent representation of a graphical image, a **logical palette**, dimensions, and other information.

**Color Correction:** Altering the colors in an image in order to print or display it such that the colors correctly match reality.



**Color Gamut:** The entire range of colors that are available on a particular graphics output device such as a display or printer.

**Color Matching:** The conversion of a color, sent from its original **color space**, to its visually closest color in the destination **color space**. See also **Image Color Management (ICM)**.

**Color Model:** See **Color Space**.

**Color Proofing:** The process of previewing, or "proofing" colors, which were developed on one device, on a different device.

**Color Space:** A mapping of color components to a multi-dimensional 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. **Grayscale**s, however, which are combinations of only black and white, can be mapped to a two-dimensional **color space**. And if transparency is considered one of the components of a **Red Green Blue (RGB)** color, four dimensions are appropriate.

**Coordinate Space:** A space based on Cartesian coordinates, which provides a means of specifying the location of each point in the space. A two-dimensional coordinate space requires two axes that are perpendicular and equal in length. Three two-dimensional coordinate spaces are generally used to describe an output surface: **world**, **page**, and **device**. To scale device-independent output for a particular physical device, a rectangular area in the **world** or **page** coordinate space is mapped into the **device** coordinate space using a **transform**.

**Design Axis:** See **Font Axis**.

**Design Vector:** A set of specific values for the **design axes** of a **multiple master** font.

**Device Context:** A structure that defines a set of graphic objects and their associated attributes, and the graphic modes that affect output. The graphic objects include a pen for line drawing, a brush for painting and filling, a **bitmap** for copying or scrolling parts of the screen, a **palette** for defining the set of available colors, a **region** for clipping and other operations, and a path for painting and drawing operations. All of these device context properties and objects together define the environment for graphics output.

**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. See [\[MS-WMF\]](#) section 2.2.2.3 for more information.

**Device Space:** The output space for graphics **transforms**. It usually refers to the client area of an application window; however, it can also include the entire desktop, a complete window, or a page of printer or plotter paper. Physical device space dimensions vary according to the dimensions set by the display, printer, or plotter technology.

**Dithering:** A form of digital **halftoning**.

**Ducking:** A **ducking** font is one which has been designed to be short enough to fit under diacritical marks or accent marks.

**Enhanced Metafile Format (EMF) :** A file format that supports the device-independent definitions of images.

**Enhanced Metafile Format Plus Extensions (EMF+):** A file format that supports the device-independent definitions of images.

**Enhanced Metafile Format (EMF) Spool Format:** A file format that specifies a structure of **Enhanced Metafile Format (EMF)** records used for defining application and device-independent printer **spool files**.

**Font Axis:** A property of font design that can assume a linear range of values. In general, a font has multiple axes. For example, a font may define an axis for **weight**, along which range the possible values for that property.

**Font Mapper:** An operating system component that maps specified font attributes to available, installed fonts on the system.

**Gamma:** The way brightness is distributed across the intensity spectrum by a graphics device. Depending on the device, the **gamma** may have a significant effect on the way colors are perceived. Technically, **gamma** is an expression of the relationship between input voltage and resulting output intensity. A perfect linear device would have a **gamma** of 1.0; a monitor or printer typically has a **gamma** in the range of 1.8 to 2.6, which effects midrange tones.

**Gamma Correction:** An adjustment to the light intensity (brightness) of an graphics device, in order to match the output more closely to the original image.

**GDI:** The **Windows Graphics Device Interface (GDI)** is an API that is supported on 16, 32 and 64-bit versions of Windows and is used for performing graphics operations and image manipulation on logical graphics objects.

**GDI+:** The **Windows Extended Graphics Device Interface (GDI+)** is an API, which is supported on 32 and 64-bit versions of Windows, for performing graphics operations and image manipulation on logical graphics objects. The extensions to **GDI** include methods for **Bezier curves** and gradient brushes.

**Grayscale:** A continuum of shades of gray used to represent an image. Continuous-tone images, such as black-and-white photographs, use an almost unlimited number of shades of gray. Conventional computer hardware and software, however, can only represent a limited gray, typically 16 or 256. Grayscale is the process of converting a continuous-tone image to an image that a computer can manipulate.

Note that grayscale is different from **dithering**. **Dithering** simulates shades of gray by altering the density and pattern of black and white dots. In grayscale, each individual dot can have a different shade of gray.

**Halftoning:** The process of converting **grayscale** or continuous-tone graphics to a representation with a discrete number of gray or tone levels.

**ICC Color Profile:** An **International Color Consortium (ICC)**-approved color management standard for specifying the attributes of imaging devices such as scanners, digital cameras, monitors and printers so that the color of an image remains true from source to destination. A **color profile** can be embedded within the image itself.

**International Color Consortium (ICC):** A group established in 1993 by eight industry vendors for the purpose of creating, promoting and encouraging the standardization and evolution of an open, vendor-neutral, cross-platform color management system architecture and components. The outcome of this co-operation was the development of the **ICC** profile specification. Version 4 of the specification, [\[ICC\]](#), is now widely used and has recently been approved as an International Standard, ISO 15076.

**Image Color Management (ICM):** Technology that ensures that a color image, graphic, or text object is rendered as closely as possible to its original intent on any device despite differences in imaging technologies and color capabilities between devices.

**Inclusive-Inclusive:** When referring to the bounds of a rectangle that consist of two coordinates—one coordinate for one corner and the other coordinate for the opposite corner—**inclusive-inclusive** means that the coordinates are part of the rectangle. If not **inclusive-inclusive**, the coordinates are not part of the rectangle, and instead are one logical unit outside the bounds of the rectangle along both coordinate axes.

**JPEG:** Joint Photographic Experts Group (JPEG): A standard still-image format that is very popular due to its excellent compression capabilities. **JPEG** files are widely used for photographic images, but are not as well suited for compressing charts and diagrams, because text can become fuzzy. **JPEG** files use the JPEG File Interchange Format (JFIF), as specified in [\[JFIF\]](#), and file extensions are .JPG or .JFF.

**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** in order for the custom colors to appear. Thus, a logical palette allows an application to use as many colors as needed without interfering with colors displayed by other applications.

**Mapping Mode:** The unit of measure for transforming logical units into device units, and also for defining the orientation of the x and y axes of the device surface.

**Metafile:** A collection of structures that can store an image in an application-independent format. The stored image can be recreated by processing the metafile structures. Also called a **vector image**, a metafile contains a sequence of drawing commands, object definitions, and configuration settings. The commands, objects, and settings recorded in a metafile can be used to render its contents on a display, as output by a printer or plotter, stored in memory, or saved to a file or stream.

**Multiple Master:** A font technology that is a variation of Adobe's **PostScript Type 1 font** format. **Multiple master** fonts are outline fonts, so changing their size does not affect the quality of their output.

Multiple master technology supports the creation of an unlimited number of custom variations of a font, called instances, as well as the emulation of **typefaces** that might not be present on the user's system.

**OpenGL:** A software API for graphics hardware, which supports the rendering of multidimensional graphical objects. The Microsoft implementation of **OpenGL** for the Windows operating system is industry-standard graphics software with which programmers can create high-quality still and animated three-dimensional color images. See [\[OPENGL\]](#).

**OpenType:** A **Unicode**-based font technology developed by Microsoft and Adobe. It is an extension to **TrueType** and **Type 1 font** technologies; OpenType allows **PostScript** glyph definitions in addition to **TrueType** glyph definitions both to reside in a common container format.

**Packed Bitmap:** A **Device-Independent Bitmap (DIB)** in which the bit array immediately follows a [BitmapInfoHeader](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.2.

**Packed DIB:** See **Packed Bitmap**.

**Page Space:** The next logical space closer to the mapping of a physical device after **world space**. It determines the **mapping mode**. Page space is defined with device-independent units, such as millimeters or inches.

**Palette:** An array of elements, each 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.

**Panose:** A classification system for font **typefaces** that is based on certain specific visual characteristics of the font, including **weight** (emphasis) and serif style.

**Playback Device Context:** The **device\_context** that defines the current graphics state during playback of the **metafile**. Although the data in an **EMF metafile** is device-independent, playback is always associated with an output device with specific properties, such as resolution, color support, etc.

**Portable Network Graphics (PNG):** A bitmapped graphics file format, specified in [\[RFC2083\]](#), that provides advanced graphics features such as 48-bit color, **alpha** channels, built-in **gamma** and **color correction**, tight compression and the ability to display at one resolution and print at another.

**Raster Operation:** The process of combining the bits in a source **bitmap** with the bits in a destination **bitmap** and the bits in a specified pattern, to achieve a desired graphical output.

**Region:** A graphics object that is non-rectilinear in shape and is defined by an array of rectangles.

**Red Green Blue (RGB):** An **additive color model** in which red, green and blue are combined in various ways to reproduce other colors.

**Stereoscopic:** The property of an image that gives the illusion of depth, as if the image were three-dimensional.

**Stock Object:** A predefined graphics object. Stock objects are standard, commonly-used objects, such as a black brush and pen. The set of pre-defined stock objects is specified in the [StockObject enumeration \(section 2.1.31\)](#).

**System Palette:** The **Palette** that is actually in use on an output device such as a display terminal. The structure of a **system palette** and the format of colors are device-dependent. Contrast with a **logical palette**.

**Transform, or Transformation:** An algorithm that transforms the size, orientation, and shape of objects that are copied from one **coordinate space** into another. Although a transform affects an object as a whole, it is applied to each point, or to each line, in the object.

**TrueType:** A scalable font technology that renders fonts for both the printer and the screen. Originally developed by Apple, it was enhanced jointly by Apple and Microsoft. Each **TrueType** font contains its own algorithms for converting printer outlines into screen **bitmaps**, which means both the outline and **bitmap** information is rasterized from the same font data. The lower-level language embedded within the **TrueType** font allows great flexibility in their design. Both **TrueType** and **Type 1 font** technologies are part of the **OpenType** format.

**Type 1 Font:** A public, standard, type format originally developed by Adobe for use with **PostScript** printers. **Type 1 fonts** contain two components—the outline font, used for printing; and the **bitmap** font set, used for screen display.

**Typeface:** The terms **typeface** and font are used interchangeably. However, a **typeface** is the primary design of a set of printed characters such as **Courier**, **Helvetica**, and **Times Roman**. A font is the particular implementation and variation of the **typeface** such as normal, bold, or italics. The distinguishing characteristic of a **typeface** is often the presence or absence of serifs.

**Vector Image:** See **metafile**.

**Weight:** A property of a font that specifies the degree of emphasis or boldness of the characters.

**Windows Color System (WCS):** The color management scheme that is used by Windows Vista. The **WCS** color management scheme is a superset of **Image Color Management (ICM)** APIs and functionality.

**Windows Graphics Device Interface (GDI):** See **GDI**.

**Windows Graphics Device Interface Plus Extensions (GDI+):** See **GDI+**.

**Windows Metafile Format (WMF):** A file format used by Windows that supports the definition of graphical images. See [MS-WMF] for more information.

**World Space:** The most abstract logical **coordinate space** for graphics **transforms**.

**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](mailto: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.

[ICC] International Color Consortium, "Image Technology Colour Management - Architecture, Profile Format, and Data Structure", Specification ICC.1:2004-10, May 2006, [http://www.color.org/icc\\_specs2.xalter](http://www.color.org/icc_specs2.xalter)

[IEC-RGB] International Electrotechnical Commission, "Colour Measurement and Management in Multimedia Systems and Equipment - Part 2-1: Default RGB Colour Space - sRGB", May 1998, <http://www.colour.org/tc8-05/Docs/colorspace/61966-2-1.pdf>

[JFIF] Hamilton, E., "JPEG File Interchange Format, Version 1.02", September 1992, <http://www.w3.org/Graphics/JPEG/jfif.txt>

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

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

[MS-WMF] Microsoft Corporation, "[Windows Metafile Format Specification](#)", June 2007.

[RFC2083] Boutell, T., et al., "PNG (Portable Network Graphics) Specification Version 1.0", RFC 2083, March 1997, <http://www.ietf.org/rfc/rfc2083.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>

[RFC2781] Hoffman, P. and Yergeau, F., "UTF-16, an encoding of ISO 10646", RFC 2781, February 2000, <http://www.ietf.org/rfc/rfc2781.txt>

[UNICODE] The Unicode Consortium, "Unicode Home Page", 2006, <http://www.unicode.org/>

[W3C-PNG] World Wide Web Consortium, "Portable Network Graphics (PNG) Specification, Second Edition", November 2003, <http://www.w3.org/TR/PNG>

### 1.2.2 Informative References

[MS-EMFPLUS] Microsoft Corporation, "[Enhanced Metafile Format Plus Extensions Specification](#)", June 2007.

[MS-EMFSPool] Microsoft Corporation, "[Enhanced Metafile Pool Format Specification](#)", July 2007.

[MSDN-WRLDPGSPC] Microsoft Corporation, "World-Space to Page-Space Transformations", <http://msdn2.microsoft.com/en-us/library/ms532657.aspx>

[OPENGL] Segal, M. and Akeley, K., "The OpenGL Graphics System: A Specification, Version 2.1", December 2006, <http://www.opengl.org/registry/doc/glspec21.20061201.pdf>

[WGFx] Yuan, F., "Windows Graphics Programming - Win32 GDI and DirectDraw", Prentice Hall PTR, 2000, ISBN: 0130869856.

If you have any trouble finding [WGFx], please check [here](#).

## 1.3 Structure Overview (Synopsis)

### 1.3.1 Metafile Structure

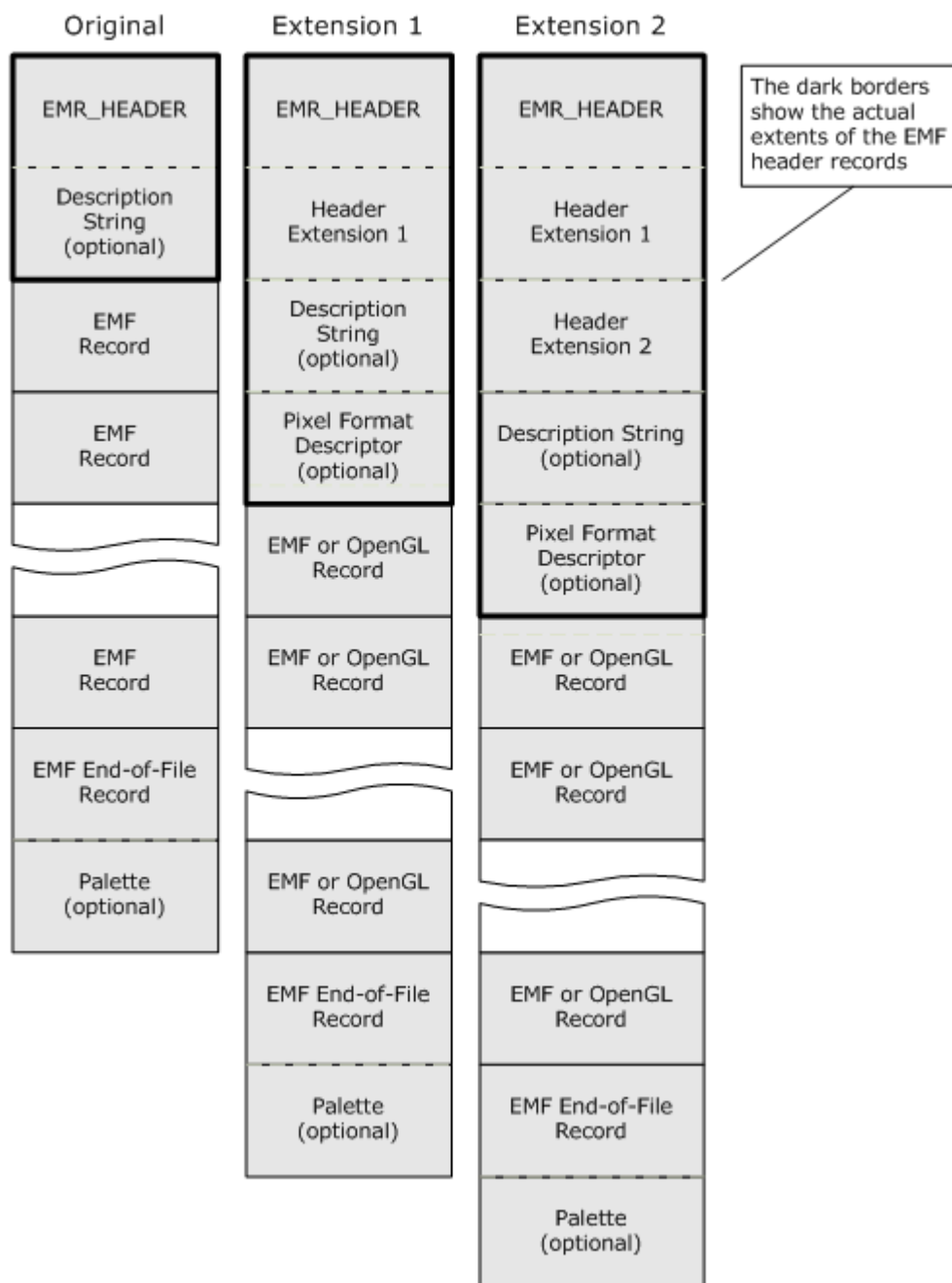
The Enhanced Metafile Format (EMF) specifies a collection of data records that contain graphics drawing commands and object definitions. EMF metafiles provide true device independence; the image defined in an EMF metafile maintains its dimensions, shape, and proportions on any output device, including printers, plotters, desktops, and in the client areas of many applications.

An EMF metafile consists of a sequence of EMF records. The first record in the metafile is always an [EMF header](#) record, and the last is always an [EMF end-of-file](#) record. Between these are records that specify drawing operations, the configuration of properties, and the creation of graphics objects, all of which together compose a device-independent picture.

The different versions of EMF are: [<1>](#)

- **Original:** This is the first version of EMF, which supports device-independent drawing commands and objects. [<2>](#)
- **Extension 1:** The first extension to EMF adds a pixel format record and support for **OpenGL** commands, enhancing the device-independence and flexibility of EMF metafiles. [<3>](#)
- **Extension 2:** The second extension to EMF adds the capability to measure distances on device surfaces in micrometers, enhancing the resolution and scalability supported by EMF metafiles. [<4>](#)

The following diagram qualitatively depicts the structures of the different versions of EMF metafiles.



**Figure 1: Enhanced Metafile Format metafile structures**

Thus, all EMF metafile can be considered to have three sections:

- **EMF Header:** The metafile header with extensions that correspond to this type of EMF metafile. The header record contains information concerning the structure and contents of the metafile, including an optional description string and pixel format descriptor. See section [2.3.4.2](#) for details concerning the EMF header.



- **EMF Records:** An array of EMF records that contain drawing orders, graphics state information, and graphics object definitions. At least one record must be present— not counting the EMF header or EMF end-of-file records—or the metafile is invalid. See sections [2.3.10](#) and [2.3.5](#) for specifications of all EMF record types.
- **EMF End-of-File:** The EMF end-of-file record signals the end of all EMF metafiles. It must be the last record in the metafile. If the EMF metafile contains an optional **palette**, it is in the form of an array of [LogPaletteEntry \(section 2.2.16\)](#) objects located in the EMF end-of-file record. Offsets to the palette are present in both the EMF header and EMF end-of-file records.

Within the portions of these files identified as EMF header records, the description and pixel format substructures are optional, and they can be placed in any order. Their presence, location, and relative order in a given EMF file are determined by offset values in the EMF header record. If present, they MUST be part of the EMF header record, and MUST NOT be located at some arbitrary location in the EMF metafile, such as between EMF records.

EMF records must be contiguous; this is required, because the information that is available for traversing the file from record to record depends on it. That is, from any given EMF record, including the header record, in order to move to the next sequential record in the file, the length of the record must be used.

### 1.3.2 Graphics Objects

Graphics objects, which are used in the drawing and painting operations specified in the records of a Enhanced Metafile Format metafile, are created by [Object Creation record types](#), specified in section [2.3.7](#), prior to the records that specify their use. These objects are designed to be reusable during the course of processing the EMF metafile.

Throughout this specification, it is assumed that these previously-defined, reusable graphics objects are available when needed for the processing of particular metafile records. This store of available objects is referred to in the text as the [EMF Object Table](#), which is described in section [3.1.1.1](#). The exact characteristics of an object store for EMF objects can determined by the particular implementation that parses or writes EMF metafiles.

The types of reusable objects that can be created and managed during EMF metafile playback include:

- Brushes, specified in section [2.2.10](#);
- Color spaces, specified in [\[MS-WMF\]](#) sections [2.2.2.5](#) and [2.2.2.6](#);
- Fonts, specified in sections [2.2.11](#), [2.2.12](#), [2.2.13](#), and [2.2.14](#);
- Palettes, specified in section [2.2.15](#); and
- Pens, specified in sections [2.2.17](#) and [2.2.18](#).

When one of these objects is created, it is assigned a 32-bit index in the EMF Object Table. An object can be "activated" by an [EMR\\_SELECTOBJECT](#) record that refers to the index it was assigned.

There are also "**stock objects**" that can be selected for use in graphics operations. Stock objects are also assigned 32-bit indexes, but not at run time—their indexes are predefined. They are distinguished from the indexes of dynamically-created graphics objects by having the most-significant bit set to 1. The other 31 bits of the index define the particular stock object, according to the [StockObject](#) enumeration, specified in section **2.1.30**.



### 1.3.3 Byte Ordering

Data in metafile records are stored in **little-endian** format.

Some computer architectures number bytes in a binary word from left to right, which is referred to as **big-endian**. The bit diagram for this documentation is big-endian. Other architectures number the bytes in a binary word from right to left, which is referred to as little-endian. The underlying file format enumerations, objects and records are little-endian.

Using the big-endian and little-endian methods, the number 0x12345678 would be stored as shown below.

Byte order	Byte 0	Byte 1	Byte 2	Byte 3
Big-endian	0x12	0x34	0x56	0x78
Little-endian	0x78	0x56	0x34	0x12

### 1.4 Relationship to Protocols and Other Structures

This document describes the Enhanced Metafile Format (EMF), which is related to the following file formats:

- [Windows Metafile Format \(WMF\)](#), as specified in [MS-WMF] uses similar graphics, commands, and objects, but **WMF** files are not device-independent.
- [Enhanced Metafile Spool Format](#), as specified in [MS-EMFSPOOL], is an application of EMF for **print job** spooling.
- [Enhanced Metafile Format Plus Extensions \(EMF+\)](#), as described in [MS-EMFPLUS], specifies object-oriented structures that can be embedded in EMF metafiles. [<5>](#)

### 1.5 Applicability Statement

Files that adhere to the Enhanced Metafile Format (EMF) metafile format can be used as portable, device-independent containers for images. The graphics supported in EMF metafiles are applicable to document content representation, including printing and plotting.

### 1.6 Versioning and Localization

This document covers versioning issues in the following areas:

**Versioning:** The enhanced metafile format has been revised twice. The different versions of Enhanced Metafile Format are:

- Original: The first version of enhanced metafile format, supporting records that define drawing commands and graphics objects. [<6>](#)
- Extension 1: Added support for OpenGL records and an optional internal pixel format descriptor. [<7>](#)
- Extension 2: Added the capability of measuring display dimensions in micrometers. [<8>](#)

**Localization:** This structure defines no locale-specific processes or data.

## 1.7 Vendor-Extensible Fields

Enhanced Metafile Format (EMF) metafiles define a mechanism for the encapsulation of arbitrary vendor-defined data. The EMF record type [EMR\\_COMMENT](#) can contain arbitrary private data that is unknown to EMF. This data is meaningful only to applications that know the format of the data and how to use it.

## 2 Structures

The following sections specify various types of Enhanced Metafile Format metafile records and enumerations.

**Note** All character strings specified in this section are encoded in **Unicode UTF16-LE** format, as specified in [\[UNICODE\]](#), unless stated otherwise.

### 2.1 EMF Enumerations

#### 2.1.1 RecordType Enumeration

The Enhanced Metafile Format **RecordType** enumeration defines values that uniquely identify **EMF** records. These values are provided in the **Type** field of each record.

```
typedef enum
{
    EMR_HEADER = 0x00000001,
    EMR_POLYBEZIER = 0x00000002,
    EMR_POLYGON = 0x00000003,
    EMR_POLYLINE = 0x00000004,
    EMR_POLYBEZIERTO = 0x00000005,
    EMR_POLYLINETO = 0x00000006,
    EMR_POLYPOLYLINE = 0x00000007,
    EMR_POLYPOLYGON = 0x00000008,
    EMR_SETWINDOWEXTTEX = 0x00000009,
    EMR_SETWINDOWORGEX = 0x0000000A,
    EMR_SETVIEWPORTEXTTEX = 0x0000000B,
    EMR_SETVIEWPORTORGEX = 0x0000000C,
    EMR_SETBRUSHORGEX = 0x0000000D,
    EMR_EOF = 0x0000000E,
    EMR_SETPIXELV = 0x0000000F,
    EMR_SETMAPPERFLAGS = 0x00000010,
    EMR_SETMAPMODE = 0x00000011,
    EMR_SETBKMODE = 0x00000012,
    EMR_SETPOLYFILLMODE = 0x00000013,
    EMR_SETROP2 = 0x00000014,
    EMR_SETSTRETCHBLTMODE = 0x00000015,
    EMR_SETTEXTALIGN = 0x00000016,
    EMR_SETCOLORADJUSTMENT = 0x00000017,
    EMR_SETTEXTCOLOR = 0x00000018,
    EMR_SETBKCOLOR = 0x00000019,
    EMR_OFFSETCLIPRGN = 0x0000001A,
    EMR_MOVETOEX = 0x0000001B,
    EMR_SETMETARGN = 0x0000001C,
    EMR_EXCLUDECLIPRECT = 0x0000001D,
    EMR_INTERSECTCLIPRECT = 0x0000001E,
    EMR_SCALEVIEWPORTEXTTEX = 0x0000001F,
    EMR_SCALEWINDOWEXTTEX = 0x00000020,
    EMR_SAVEDC = 0x00000021,
    EMR_RESTOREDC = 0x00000022,
    EMR_SETWORLDTRANSFORM = 0x00000023,
    EMR_MODIFYWORLDTRANSFORM = 0x00000024,
    EMR_SELECTOBJECT = 0x00000025,
    EMR_CREATEPEN = 0x00000026,
    EMR_CREATEBRUSHINDIRECT = 0x00000027,
    EMR_DELETEOBJECT = 0x00000028,
```

EMR\_ANGLEARC = 0x00000029,  
EMR\_ELLIPSE = 0x0000002A,  
EMR\_RECTANGLE = 0x0000002B,  
EMR\_ROUNDRECT = 0x0000002C,  
EMR\_ARC = 0x0000002D,  
EMR\_CHORD = 0x0000002E,  
EMR\_PIE = 0x0000002F,  
EMR\_SELECTPALETTE = 0x00000030,  
EMR\_CREATEPALETTE = 0x00000031,  
EMR\_SETPALETTEENTRIES = 0x00000032,  
EMR\_RESIZEPALETTE = 0x00000033,  
EMR\_REALIZEPALETTE = 0x00000034,  
EMR\_EXTFLOODFILL = 0x00000035,  
EMR\_LINETO = 0x00000036,  
EMR\_ARCTO = 0x00000037,  
EMR\_POLYDRAW = 0x00000038,  
EMR\_SETARCDIRECTION = 0x00000039,  
EMR\_SETMITERLIMIT = 0x0000003A,  
EMR\_BEGINPATH = 0x0000003B,  
EMR\_ENDPATH = 0x0000003C,  
EMR\_CLOSEFIGURE = 0x0000003D,  
EMR\_FILLPATH = 0x0000003E,  
EMR\_STROKEANDFILLPATH = 0x0000003F,  
EMR\_STROKEPATH = 0x00000040,  
EMR\_FLATTENPATH = 0x00000041,  
EMR\_WIDENPATH = 0x00000042,  
EMR\_SELECTCLIPPATH = 0x00000043,  
EMR\_ABORTPATH = 0x00000044,  
EMR\_RESERVED\_69 = 0x00000045,  
EMR\_COMMENT = 0x00000046,  
EMR\_FILLRGN = 0x00000047,  
EMR\_FRAMERGN = 0x00000048,  
EMR\_INVERTRGN = 0x00000049,  
EMR\_PAINTRGN = 0x0000004A,  
EMR\_EXTSELECTCLIPRGN = 0x0000004B,  
EMR\_BITBLT = 0x0000004C,  
EMR\_STRETCHBLT = 0x0000004D,  
EMR\_MASKBLT = 0x0000004E,  
EMR\_PLGBLT = 0x0000004F,  
EMR\_SETDIBITSTODEVICE = 0x00000050,  
EMR\_STRETCHDIBITS = 0x00000051,  
EMR\_EXTCREATEFONTINDIRECTW = 0x00000052,  
EMR\_EXTTEXTOUTA = 0x00000053,  
EMR\_EXTTEXTOUTW = 0x00000054,  
EMR\_POLYBEZIER16 = 0x00000055,  
EMR\_POLYGON16 = 0x00000056,  
EMR\_POLYLINE16 = 0x00000057,  
EMR\_POLYBEZIERTO16 = 0x00000058,  
EMR\_POLYLINETO16 = 0x00000059,  
EMR\_POLYPOLYLINE16 = 0x0000005A,  
EMR\_POLYPOLYGON16 = 0x0000005B,  
EMR\_POLYDRAW16 = 0x0000005C,  
EMR\_CREATEMONOBRUSH = 0x0000005D,  
EMR\_CREATEDIBPATTERNBRUSHPT = 0x0000005E,  
EMR\_EXTCREATEPEN = 0x0000005F,  
EMR\_POLYTEXTOUTA = 0x00000060,  
EMR\_POLYTEXTOUTW = 0x00000061,  
EMR\_SETICMMode = 0x00000062,

```

EMR_CREATECOLORSPACE = 0x00000063,
EMR_SETCOLORSPACE = 0x00000064,
EMR_DELETECOLORSPACE = 0x00000065,
EMR_GLSRECORD = 0x00000066,
EMR_GLSBOUNDEDRECORD = 0x00000067,
EMR_PIXELFORMAT = 0x00000068,
EMR_DRAWESCAPE = 0x00000069,
EMR_EXTESCAPE = 0x0000006A,
EMR_RESERVED_107 = 0x0000006B,
EMR_SMALLTEXTOUT = 0x0000006C,
EMR_FORCEUFIMAPPING = 0x0000006D,
EMR_NAMEDESCAPE = 0x0000006E,
EMR_COLORCORRECTPALETTE = 0x0000006F,
EMR_SETICMPROFILEA = 0x00000070,
EMR_SETICMPROFILEW = 0x00000071,
EMR_ALPHABLEND = 0x00000072,
EMR_SETLAYOUT = 0x00000073,
EMR_TRANSPARENTBLT = 0x00000074,
EMR_RESERVED_117 = 0x00000075,
EMR_GRADIENTFILL = 0x00000076,
EMR_SETLINKEDUFIS = 0x00000077,
EMR_SETTEXTJUSTIFICATION = 0x00000078,
EMR_COLORMATCHTOTARGETW = 0x00000079,
EMR_CREATECOLORSPACEW = 0x0000007A
} RecordType;

```

**EMR\_HEADER:** This record defines the start of the metafile and specifies its characteristics; its contents, including the dimensions of the embedded image; the number of records in the metafile; and the resolution of the device on which the embedded image was created. These values make it possible for the metafile to be device-independent.

**EMR\_POLYBEZIER:** This record defines one or more **Bezier curves**. Cubic Bezier curves are defined using specified endpoints and control points, and are stroked with the current pen.

**EMR\_POLYGON:** This record defines a polygon consisting of two or more vertexes connected by straight lines. The polygon is outlined by using the current pen and filled by using the current brush and polygon fill mode. The polygon is closed automatically by drawing a line from the last vertex to the first.

**EMR\_POLYLINE:** This record defines a series of line segments by connecting the points in the specified array.

**EMR\_POLYBEZIERTO:** This record defines one or more Bezier curves based upon the current position.

**EMR\_POLYLINETO:** This record defines one or more straight lines based upon the current position. A line is drawn from the current position to the first point specified by the points field by using the current pen. For each additional line, drawing is performed from the ending point of the previous line to the next point specified by points.

**EMR\_POLYPOLYLINE:** This record defines multiple series of connected line segments. The line segments are drawn by using the current pen. The figures formed by the segments are not filled. The current position is neither used nor updated by this record.

**EMR\_POLYPOLYGON:** This record defines a series of closed polygons. Each polygon is outlined by using the current pen and filled by using the current brush and polygon fill mode. The polygons defined by this record can overlap.

**EMR\_SETWINDOWEXT:** This record defines the window extent.

**EMR\_SETWINDOWORG:** This record defines the window origin.

**EMR\_SETVIEWPORTEXT:** This record defines the viewport extent.

**EMR\_SETVIEWPORTORG:** This record defines the viewport origin.

**EMR\_SETBRUSHORG:** This record defines the origin of the current brush.

**EMR\_EOF:** This record indicates the end of the metafile.

**EMR\_SETPIXELV:** This record defines the color of the pixel at the specified logical coordinates.

**EMR\_SETMAPPERFLAGS:** This record redefines the algorithm the **font mapper** uses when it maps logical fonts to physical fonts.

**EMR\_SETMAPMODE:** This record defines the **mapping mode** of the **playback device context**. The mapping mode defines the unit of measure used to transform page-space units into device-space units, and also defines the orientation of the device's x and y axes.

**EMR\_SETBKMODE:** This record defines the background mix mode of the playback device context. The background mix mode is used with text, hatched brushes, and pen styles that are not solid lines.

**EMR\_SETPOLYFILLMODE:** This record defines polygon fill mode.

**EMR\_SETROP2:** This record defines binary **raster operation** mode.

**EMR\_SETSTRETCHBLTMODE:** This record defines **bitmap** stretch mode.

**EMR\_SETTEXTALIGN:** This record defines text alignment.

**EMR\_SETCOLORADJUSTMENT:** This record defines the color adjustment values for the playback device context using the specified values.

**EMR\_SETTEXTCOLOR:** This record defines the current text color.

**EMR\_SETBKCOLOR:** This record defines the background color.

**EMR\_OFFSETCLIPRGN:** This record redefines the clipping **region** of the playback device context by the specified offsets.

**EMR\_MOVETOEX:** This record defines coordinates of the new current position in logical units.

**EMR\_SETMETARGN:** This record intersects the current clipping region for the playback device context with the current metaregion and saves the combined region as the new metaregion. The clipping region is reset to a null region.

**EMR\_EXCLUDECLIPRECT:** This record defines a new clipping region that consists of the existing clipping region minus the specified rectangle.

**EMR\_INTERSECTCLIPRECT:** This record defines a new clipping region from the intersection of the current clipping region and the specified rectangle.

**EMR\_SCALEVIEWPORTEXT:** This record redefines the viewport for the playback device context using the ratios formed by the specified multiplicands and divisors.

**EMR\_SCALEWINDOWEXT:** This record redefines the window for the playback device context using the ratios formed by the specified multiplicands and divisors.

**EMR\_SAVEDC:** This record saves the current state of the playback device context by copying data describing selected objects and graphic modes—including the bitmap, brush, palette, font, pen, region, drawing mode, and mapping mode—to a stack of saved device contexts.

**EMR\_RESTOREDC:** This record restores the playback device context to the specified saved state. The playback device context is restored by popping state information off a stack of saved device contexts created by earlier [EMR\\_SAVEDC](#) records.

**EMR\_SETWORLDTRANSFORM:** This record defines a two-dimensional linear transformation between world space and page space—for more information, see [\[MSDN-WRLDPGSPC\]](#)—for the playback device context. This transformation can be used to scale, rotate, shear, or translate graphics output.

**EMR\_MODIFYWORLDTRANSFORM:** This record redefines the world transformation for the playback device context using the specified mode.

**EMR\_SELECTOBJECT:** This record adds an object to the playback device context, identifying it by its index in the [EMF Object Table](#).

**EMR\_CREATEPEN:** This record defines a logical pen that has the specified style, width, and color. The pen can subsequently be selected into the playback device context and used to draw lines and curves.

**EMR\_CREATEBRUSHINDIRECT:** This record defines a logical brush that has the specified style, color, and pattern. The **BrushStyle** MUST be **BS\_SOLID**, **BS\_HATCHED**, or **BS\_NULL**.[<9>](#) The brush can subsequently be selected into the playback device context and used in graphics operations.

**EMR\_DELETEOBJECT:** This record deletes a graphics object, clearing its index in the EMF Object Table. If the deleted object is selected in the playback device context, the default object for that context property MUST be restored.

**EMR\_ANGLEARC:** This record defines a line segment of an arc. The line segment is drawn from the current position to the beginning of the arc. The arc is drawn along the perimeter of a circle with the given radius and center. The length of the arc is defined by the given start and sweep angles.

**EMR\_ELLIPSE:** This record defines an ellipse. The center of the ellipse is the center of the specified bounding rectangle. The ellipse is outlined by using the current pen and is filled by using the current brush.

**EMR\_RECTANGLE:** This record defines a rectangle. The rectangle is outlined by using the current pen and filled by using the current brush.

**EMR\_ROUNDRECT:** This record defines a rectangle with rounded corners. The rectangle is outlined by using the current pen and filled by using the current brush.

**EMR\_ARC:** This record defines an elliptical arc.

**EMR\_CHORD:** This record defines a chord (a region bounded by the intersection of an ellipse and a line segment, called a secant). The chord is outlined by using the current pen and filled by using the current brush.

**EMR\_PIE:** This record defines a pie-shaped wedge bounded by the intersection of an ellipse and two radials. The pie is outlined by using the current pen and filled by using the current brush.

**EMR\_SELECTPALETTE:** This record adds a [LogPalette](#) object to the playback device context, identifying it by its index in the EMF Object Table.

**EMR\_CREATEPALETTE:** This record defines a LogPalette object.

**EMR\_SETPALETTEENTRIES:** This record defines **RGB** (red, green, blue) color values in a range of entries in a LogPalette object.

**EMR\_RESIZEPALETTE:** This record increases or decreases the size of a **logical palette**.

**EMR\_REALIZEPALETTE:** This record maps entries from the current logical palette to the **system palette**.

**EMR\_EXTFLOODFILL:** This record fills an area of the display surface with the current brush.

**EMR\_LINETO:** This record defines a line from the current position up to, but not including, the specified point. It resets the current position to the specified point.

**EMR\_ARCTO:** This record defines an elliptical arc. It resets the current position to the end point of the arc.

**EMR\_POLYDRAW:** This record defines a set of line segments and Bezier curves.

**EMR\_SETARCDIRECTION:** This record defines the drawing direction to be used for arc and rectangle operations.

**EMR\_SETMITERLIMIT:** This record defines the limit for the length of miter joins for the playback device context.

**EMR\_BEGINPATH:** This record opens a path bracket in the playback device context.

**EMR\_ENDPATH:** This record closes a path bracket and selects the path defined by the bracket into the playback device context.

**EMR\_CLOSEFIGURE:** This record closes an open figure in a path.

**EMR\_FILLPATH:** This record closes any open figures in the current path and fills the path's interior by using the current brush and polygon-filling mode.

**EMR\_STROKEANDFILLPATH:** This record closes any open figures in a path, strokes the outline of the path by using the current pen, and fills its interior by using the current brush.

**EMR\_STROKEPATH:** This record renders the specified path by using the current pen.

**EMR\_FLATTENPATH:** This record transforms any curve in the path that is selected into the playback device context, turning each curve into a sequence of lines.

**EMR\_WIDENPATH:** This record redefines the current path as the area that would be painted if the path were stroked using the pen currently selected into the playback device context.



**EMR\_SELECTCLIPPATH:** This record defines the current path as a clipping region for the playback device context, combining the new region with any existing clipping region using the specified mode.

**EMR\_ABORTPATH:** This record aborts a path bracket or discards the path from a closed path bracket.

**EMR\_RESERVED\_69:** This record type is reserved and MUST NOT be used.

**EMR\_COMMENT:** This record contains public or private data. The public data can be used to add new or special-purpose commands to the suite of metafile records. The private data is unknown to EMF; it is meaningful only to applications that know the format of the data and how to use it. Application-specific data in [EMR\\_COMMENT](#) records is ignored during processing of EMF metafiles, except to hand it off to the application that understands it

**EMR\_FILLRGN:** This record fills the specified region by using the specified brush.

**EMR\_FRAMERGN:** This record draws a border around the specified region using the specified brush.

**EMR\_INVERTRGN:** This record inverts the colors in the specified region.

**EMR\_PAINTRGN:** This record paints the specified region by using the brush currently selected into the playback device context.

**EMR\_EXTSELECTCLIPRGN:** This record combines the specified region with the current clip region using the specified mode.

**EMR\_BITBLT:** This record performs a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern, to the playback device context, according to a specified raster operation.

**EMR\_STRETCHBLT:** This record performs a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern, to the playback device context, according to a specified raster operation, stretching or compressing the output to fit the dimensions of the destination, if necessary.

**EMR\_MASKBLT:** This record performs a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern and with the application of a color mask bitmap, to the playback device context, according to specified foreground and background raster operations.

**EMR\_PLGBLT:** This record performs a block transfer of pixels from a source bitmap, with the application of a color mask bitmap, to a parallelogram in the playback device context.

**EMR\_SETDIBITSTODEVICE:** This record performs a block transfer of pixels from specified scanlines of a source bitmap to the playback device context.

**EMR\_STRETCHDIBITS:** This record performs a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern, to the playback device context, according to a specified raster operation, stretching or compressing the output to fit the dimensions of the destination, if necessary.

**EMR\_EXTCREATEFONTINDIRECTW:** This record defines a logical font that has the specified characteristics. The font can subsequently be selected as the current font for the playback device context.

**EMR\_EXTTEXTOUTA:** This record defines **ASCII** text using the currently selected font, background color, and text color. An optional rectangle MAY be used for clipping, opaquing, or both.

**EMR\_EXTTEXTOUTW:** This record defines Unicode text, as specified in [\[UNICODE\]](#), by using the currently selected font, background color, and text color. An optional rectangle MAY be used for clipping, opaquing, or both.

**EMR\_POLYBEZIER16:** This record defines one or more Bezier curves. The curves are drawn using the current pen.

**EMR\_POLYGON16:** This record defines a polygon consisting of two or more vertexes connected by straight lines. The polygon is outlined by using the current pen and filled by using the current brush and polygon fill mode. The polygon is closed automatically by drawing a line from the last vertex to the first.

**EMR\_POLYLINE16:** This record defines a series of line segments by connecting the points in the specified array.

**EMR\_POLYBEZIERTO16:** This record defines one or more Bezier curves based on the current position.

**EMR\_POLYLINETO16:** This record defines one or more straight lines based upon the current position. A line is drawn from the current position to the first point specified by the **Points** field by using the current pen. For each additional line, drawing is performed from the ending point of the previous line to the next point specified by **Points**.

**EMR\_POLYPOLYLINE16:** This record defines multiple series of connected line segments.

**EMR\_POLYPOLYGON16:** This record defines a series of closed polygons. Each polygon is outlined by using the current pen and filled by using the current brush and polygon fill mode. The polygons specified by this record can overlap.

**EMR\_POLYDRAW16:** This record defines a set of line segments and Bezier curves.

**EMR\_CREATEMONOBRUSH:** This record defines a logical brush with the specified bitmap pattern. The bitmap can be a **DIB** section bitmap or it can be a device-dependent bitmap.

**EMR\_CREATEDIBPATTERNBRUSHPT:** This record defines logical brush that has the pattern specified by the device-independent bitmap (DIB).

**EMR\_EXTCREATEPEN:** This record defines a logical cosmetic or geometric pen that has the specified style, width, and brush attributes.

**EMR\_POLYTEXTOUTA:** This record defines several ASCII strings using the font and text colors currently selected in the playback device context.

**EMR\_POLYTEXTOUTW:** This record defines several Unicode strings using the font and text colors currently selected in the playback device context.

**EMR\_SETICMMODE:** This record defines **Image Color Management (ICM)** to be enabled, disabled, or queried on the playback device context.

**EMR\_CREATECOLORSPACE:** This record defines a logical **Color Space**.

**EMR\_SETCOLORSPACE:** This record sets current logical Color Space.

**EMR\_DELETECOLORSPACE:** This record removes a [LogColorSpace](#) or [LogColorSpaceW](#) object from the playback device context, identifying it by its index in the EMF Object Table.

**EMR\_GLSRECORD:** This record defines an enhanced metafile record generated by OpenGL functions. It contains data for OpenGL functions that scale automatically to the OpenGL viewport.

**EMR\_GLSBOUNDEDRECORD:** This record defines an enhanced metafile record generated by OpenGL functions within a bounding rectangle. It contains data for OpenGL functions with information in pixel units that **MUST** be scaled when playing the metafile.

**EMR\_PIXELFORMAT:** This record sets the pixel format of the playback device context to the format specified by the data in this record.

**EMR\_DRAWESCAPE:** This record passes arbitrary information to the driver. The intent is that the information will result in drawing being done.

**EMR\_EXTESCAPE:** This record passes arbitrary information to the driver. The intent is that the information will not result in drawing being done.

**EMR\_RESERVED\_107:** This record type is reserved and **MUST NOT** be used.

**EMR\_SMALLTEXTOUT:** This record outputs a string.

**EMR\_FORCEUFIMAPPING:** This record forces the font mapper to match fonts based on their **UniversalFontId** in preference to their **LogFont** information.

**EMR\_NAMEDESCAPE:** This record passes arbitrary information to the given named driver.

**EMR\_COLORCORRECTPALETTE:** This record defines how to correct the entries of a palette using the **WCS 1.0** fields in the playback device context.

**EMR\_SETICMPROFILEA:** This record defines how to set a specified **color profile** as the output profile for the playback device context using an ASCII filename.

**EMR\_SETICMPROFILEW:** This record defines how to set a specified color profile as the output profile for the playback device context using a Unicode filename.

**EMR\_ALPHABLEND:** This record performs a block transfer of pixels from a source bitmap, including **alpha transparency** data, to the playback device context, according to a specified blending operation.

**EMR\_SETLAYOUT:** This record redefines the layout of the playback device context.

**EMR\_TRANSPARENTBLT:** This record performs a block transfer of pixels from a source bitmap, treating a specified color as transparent, to the playback device context, stretching or compressing the output to fit the dimensions of the destination, if necessary.

**EMR\_RESERVED\_117:** This record type is reserved and **MUST NOT** be used.

**EMR\_GRADIENTFILL:** This record defines how to fill rectangle and triangle records.

**EMR\_SETLINKEDUFIS:** This record sets the **UniversalFontIds** of linked fonts to use during character lookup.

**EMR\_SETTEXTJUSTIFICATION:** This record sets the amount of extra space to add to break characters for justification purposes.

**EMR\_COLORMATCHTOTARGETW:** This record defines how to preview colors as they would appear on the target device.

**EMR\_CREATECOLORSPACEW:** This record defines a logical Color Space using a Unicode file name.

### 2.1.2 ArcDirection Enumeration

The Enhanced Metafile Format (EMF) **ArcDirection** enumeration is used in setting the drawing direction for arc and rectangle output.

```
typedef enum
{
    AD_COUNTERCLOCKWISE = 0x0001,
    AD_CLOCKWISE = 0x0002
} ArcDirection;
```

**AD\_COUNTERCLOCKWISE:** Figures drawn counterclockwise.

**AD\_CLOCKWISE:** Figures drawn clockwise.

### 2.1.3 ArmStyle Enumeration

The Enhanced Metafile Format **ArmStyle** enumeration defines values for one of the characteristics in the **Panose** system for classifying **typefaces**.

```
typedef enum
{
    PAN_ANY = 0x00,
    PAN_NO_FIT = 0x01,
    PAN_STRAIGHT_ARMS_HORZ = 0x02,
    PAN_STRAIGHT_ARMS_WEDGE = 0x03,
    PAN_STRAIGHT_ARMS_VERT = 0x04,
    PAN_STRAIGHT_ARMS_SINGLE_SERIF = 0x05,
    PAN_STRAIGHT_ARMS_DOUBLE_SERIF = 0x06,
    PAN_BENT_ARMS_HORZ = 0x07,
    PAN_BENT_ARMS_WEDGE = 0x08,
    PAN_BENT_ARMS_VERT = 0x09,
    PAN_BENT_ARMS_SINGLE_SERIF = 0x0A,
    PAN_BENT_ARMS_DOUBLE_SERIF = 0x0B
} ArmStyle;
```

**PAN\_ANY:** Any.

**PAN\_NO\_FIT:** No fit.

**PAN\_STRAIGHT\_ARMS\_HORZ:** Straight arms/horizontal.

**PAN\_STRAIGHT\_ARMS\_WEDGE:** Straight arms/wedge.

**PAN\_STRAIGHT\_ARMS\_VERT:** Straight arms/vertical.

**PAN\_STRAIGHT\_ARMS\_SINGLE\_SERIF:** Straight arms/single-serif.

**PAN\_STRAIGHT\_ARMS\_DOUBLE\_SERIF:** Straight arms/double-serif.

**PAN\_BENT\_ARMS\_HORZ:** Non-straight arms/horizontal.

**PAN\_BENT\_ARMS\_WEDGE:** Non-straight arms/wedge.

**PAN\_BENT\_ARMS\_VERT:** Non-straight arms/vertical.

**PAN\_BENT\_ARMS\_SINGLE\_SERIF:** Non-straight arms/single-serif.

**PAN\_BENT\_ARMS\_DOUBLE\_SERIF:** Non-straight arms/double-serif.

#### 2.1.4 BackgroundMode Enumeration

The Enhanced Metafile Format **BackgroundMode** enumeration is used to specify the background mode to be used with text, hatched brushes, and pen styles that are not solid. The background mode determines how to combine the background with foreground text, hatched brushes, and pen styles that are not solid lines.

```
typedef enum
{
    TRANSPARENT = 0x0001,
    OPAQUE = 0x0002
} BackgroundMode;
```

**TRANSPARENT:** Background remains untouched.

**OPAQUE:** Background is filled with the current background color before the text, hatched brush, or pen is drawn.

#### 2.1.5 ColorAdjustment Enumeration

The Enhanced Metafile Format **ColorAdjustment** enumeration is used to specify how the output image SHOULD be prepared when the stretch mode is **HALFTONE**.

```
typedef enum
{
    CA_NEGATIVE = 0x0001,
    CA_LOG_FILTER = 0x0002
} ColorAdjustment;
```

**CA\_NEGATIVE:** Specifies that the negative of the original image SHOULD be displayed.

**CA\_LOG\_FILTER:** Specifies that a logarithmic process SHOULD be applied to the final density of the output colors. This will increase the color contrast when the luminance is low.

#### 2.1.6 ColorMatchToTarget Enumeration

The Enhanced Metafile Format **ColorMatchToTarget** enumeration is used to determine whether a color profile has been embedded in the metafile.

```
typedef enum
{
    COLORMATCHTOTARGET_NOTEMBEDDED = 0x00000000,
    COLORMATCHTOTARGET_EMBEDDED = 0x00000001
}
```

```
} ColorMatchToTarget;
```

**COLORMATCHTOTARGET\_NOTEMBEDDED:** Indicates that a color profile has not been embedded in the metafile.

**COLORMATCHTOTARGET\_EMBEDDED:** Indicates that a color profile has been embedded in the metafile.

### 2.1.7 ColorSpace Enumeration

The Enhanced Metafile Format **ColorSpace** enumeration is used to specify when to turn **color proofing** on and off, and when to delete transforms.

```
typedef enum
{
    CS_ENABLE = 0x00000001,
    CS_DISABLE = 0x00000002,
    CS_DELETE_TRANSFORM = 0x00000003
} ColorSpace;
```

**CS\_ENABLE:** Maps colors to the target device's color gamut. This enables color proofing. All subsequent draw commands to the playback device context will render colors as they would appear on the target device.

**CS\_DISABLE:** Disables color proofing.

**CS\_DELETE\_TRANSFORM:** If color management is enabled for the target profile, disables it and deletes the concatenated transform.

### 2.1.8 Contrast Enumeration

The Enhanced Metafile Format **Contrast** enumeration defines values for one of the characteristics in the Panose system for classifying typefaces.

```
typedef enum
{
    PAN_ANY = 0x00,
    PAN_NO_FIT = 0x01,
    PAN_CONTRAST_NONE = 0x02,
    PAN_CONTRAST_VERY_LOW = 0x03,
    PAN_CONTRAST_LOW = 0x04,
    PAN_CONTRAST_MEDIUM_LOW = 0x05,
    PAN_CONTRAST_MEDIUM = 0x06,
    PAN_CONTRAST_MEDIUM_HIGH = 0x07,
    PAN_CONTRAST_HIGH = 0x08,
    PAN_CONTRAST_VERY_HIGH = 0x09
} Contrast;
```

**PAN\_ANY:** Any.

**PAN\_NO\_FIT:** No fit.

**PAN\_CONTRAST\_NONE:** None.

**PAN\_CONTRAST\_VERY\_LOW:** Very low.

**PAN\_CONTRAST\_LOW:** Low.

**PAN\_CONTRAST\_MEDIUM\_LOW:** Medium low.

**PAN\_CONTRAST\_MEDIUM:** Medium.

**PAN\_CONTRAST\_MEDIUM\_HIGH:** Medium high.

**PAN\_CONTRAST\_HIGH:** High.

**PAN\_CONTRAST\_VERY\_HIGH:** Very high.

### 2.1.9 DIBColors Enumeration

The Enhanced Metafile Format **DIBColors** enumeration defines how the values in the color table of a DIB SHOULD be interpreted.

```
typedef enum
{
    DIB_RGB_COLORS = 0x00,
    DIB_PAL_COLORS = 0x01
} DIBColors;
```

**DIB\_RGB\_COLORS:** The color table contains literal RGB values.

**DIB\_PAL\_COLORS:** The color table consists of an array of 16-bit indexes into the currently-selected [LogPalette](#) object, specified in section [2.2.15](#).

### 2.1.10 EmrComment Enumeration

The Enhanced Metafile Format (EMF) **EmrComment** enumeration defines the types of data that a public comment record can contain, as specified in section [2.3.3.4](#).

```
typedef enum
{
    EMR_COMMENT_WINDOWS_METAFILE = 0x80000001,
    EMR_COMMENT_BEGINGROUP = 0x00000002,
    EMR_COMMENT_ENDGROUP = 0x00000003,
    EMR_COMMENT_MULTIFORMATS = 0x40000004,
    EMR_COMMENT_UNICODE_STRING = 0x00000040,
    EMR_COMMENT_UNICODE_END = 0x00000080
} EmrComment;
```

**EMR\_COMMENT\_WINDOWS\_METAFILE:** This comment record contains a specification of an image in [Windows Metafile Format \(WMF\)](#). See [MS-WMF] for more information.

**EMR\_COMMENT\_BEGINGROUP:** This comment record identifies the beginning of a group of drawing records. It identifies an object within an EMF metafile.

**EMR\_COMMENT\_ENDGROUP:** This comment record identifies the end of a group of drawing records. For every [EMR\\_COMMENT\\_BEGINGROUP](#) record, an [EMR\\_COMMENT\\_ENDGROUP](#) record MUST be included in the metafile, and they MAY be nested.

**EMR\_COMMENT\_MULTIFORMATS:** This comment record allows multiple definitions of an image to be included in the metafile. Using this comment, for example, an application can include encapsulated **PostScript** text as well as an EMF definition of a image.

**EMR\_COMMENT\_UNICODE\_STRING:** This comment record is reserved and MUST NOT be used in an EMF metafile.

**EMR\_COMMENT\_UNICODE\_END:** This comment record is reserved and MUST NOT be used in an EMF metafile.

### 2.1.11 ExtTextOutOptions Enumeration

The Enhanced Metafile Format (EMF) **ExtTextOutOptions** enumeration specifies parameters that control various aspects of the output of text by [EMR\\_SMALLTEXTOUT \(section 2.3.5.37\)](#) records and in [EmrText](#) objects.

```
typedef enum
{
    ETO_OPAQUE = 0x00000002,
    ETO_CLIPPED = 0x00000004,
    ETO_GLYPH_INDEX = 0x00000010,
    ETO_RTLREADING = 0x00000080,
    ETO_NO_RECT = 0x00000100,
    ETO_SMALL_CHARS = 0x00000200,
    ETO_NUMERICSLocal = 0x00000400,
    ETO_NUMERICSLATIN = 0x00000800,
    ETO_IGNORELANGUAGE = 0x00001000,
    ETO_PDY = 0x00002000,
    ETO_REVERSE_INDEX_MAP = 0x00010000
} ExtTextOutOptions;
```

**ETO\_OPAQUE:** This bit indicates that the current background color SHOULD be used to fill the rectangle.

**ETO\_CLIPPED:** This bit indicates that the text SHOULD be clipped to the rectangle.

**ETO\_GLYPH\_INDEX:** This bit indicates that the codes for characters in an output text string are actually indexes of the character glyphs in a **TrueType** font. Glyph indexes are font-specific, so to display the correct characters on playback, the font that is used MUST be identical to the font used to generate the indexes. [<10>](#)

**ETO\_RTLREADING:** This bit indicates that the text MUST be laid out in right-to-left reading order, instead of the default left-to-right order. This SHOULD be applied only when the font selected into the playback device context is either Hebrew or Arabic.

**ETO\_NO\_RECT:** This bit indicates that the record does not specify a bounding rectangle for the text output.

**ETO\_SMALL\_CHARS:** This bit indicates that the codes for characters in an output text string are 8 bits, derived from the low bytes of 16-bit Unicode UTF16-LE character codes, in which the high byte is assumed to be 0.

**ETO\_NUMERICSLocal:** This bit indicates that to display numbers, digits appropriate to the locale SHOULD be used.



**ETO\_NUMERICSLATIN:** This bit indicates that to display numbers, European digits SHOULD be used.

**ETO\_IGNORELANGUAGE:** This bit indicates that no special operating system processing for glyph placement should be performed on right-to-left strings; that is, all glyph positioning SHOULD be taken care of by drawing and state records in the metafile. [<11>](#)

**ETO\_PDY:** This bit indicates that both horizontal and vertical character displacement values SHOULD be provided.

**ETO\_REVERSE\_INDEX\_MAP:** This bit is reserved and SHOULD NOT be used.

### 2.1.12 FamilyType Enumeration

The Enhanced Metafile Format **FamilyType** enumeration defines values for one of the characteristics in the Panose system for classifying typefaces.

```
typedef enum
{
    PAN_ANY = 0x00,
    PAN_NO_FIT = 0x01,
    PAN_FAMILY_TEXT_DISPLAY = 0x02,
    PAN_FAMILY_SCRIPT = 0x03,
    PAN_FAMILY_DECORATIVE = 0x04,
    PAN_FAMILY_PICTORIAL = 0x05
} FamilyType;
```

**PAN\_ANY:** Any.

**PAN\_NO\_FIT:** No fit.

**PAN\_FAMILY\_TEXT\_DISPLAY:** Text and display.

**PAN\_FAMILY\_SCRIPT:** Script.

**PAN\_FAMILY\_DECORATIVE:** Decorative.

**PAN\_FAMILY\_PICTORIAL:** Pictorial.

### 2.1.13 FloodFill Enumeration

The Enhanced Metafile Format **FloodFill** enumeration defines values that specify how a fill area SHOULD be calculated.

```
typedef enum
{
    FLOODFILLBORDER = 0x00,
    FLOODFILLSURFACE = 0x01
} FloodFill;
```

**FLOODFILLBORDER:** The fill area is bounded by the color specified by the **Color** field.

**FLOODFILLSURFACE:** The fill area is defined by the color that is specified by the **Color** field. Filling continues outward in all directions as long as the color is encountered. This style is useful for filling areas with multicolored boundaries.

### 2.1.14 FormatSignature Enumeration

The Enhanced Metafile Format **FormatSignature** enumeration defines values that are used to identify the format of image data in EMF records. In an [EmrFormat](#) object, specified in section [2.2.3](#), the format signature specifies how the fields in that object will be used.

```
typedef enum
{
    ENHMETA_SIGNATURE = 0x464D4520,
    EPS_SIGNATURE = 0x46535045
} FormatSignature;
```

**ENHMETA\_SIGNATURE:** The value of this member is the sequence of ASCII characters "FME ", which happens to be the reverse of the string " EMF", and it denotes data in EMF.

**Note** The space character in the string is significant and MUST be present.

This signature MUST be used in EMF metafile header records, specified in section [2.3.4.2](#), to identify the type of metafile; and it MAY be used in EmrFormat objects to specify the format of embedded graphics data in [EMR\\_COMMENT\\_MULTIFORMATS](#), specified in section [2.3.3.4.3](#).

**EPS\_SIGNATURE:** The value of this member is the sequence of ASCII characters "FSPE", which happens to be the reverse of the string "EPSF", and it denotes embedded PostScript data.

This signature MAY be used in EmrFormat objects to specify the format of embedded graphics data in EMR\_COMMENT\_MULTIFORMATS records.

### 2.1.15 GradientFill Enumeration

The Enhanced Metafile Format GradientFill enumeration defines values that specify what shapes SHOULD be used to calculate the gradient fill.

```
typedef enum
{
    GRADIENT_FILL_RECT_H = 0x00000000,
    GRADIENT_FILL_RECT_V = 0x00000001,
    GRADIENT_FILL_TRIANGLE = 0x00000002
} GradientFill;
```

**GRADIENT\_FILL\_RECT\_H:** In this mode, two endpoints describe a rectangle. The rectangle is defined to have specific colors, defined by [TriVertex](#) objects, on the left and right edges. The color SHOULD be interpolated along a gradient from the left to right edges and used to fill the interior.

**GRADIENT\_FILL\_RECT\_V:** In this mode, two endpoints describe a rectangle. The rectangle is defined to have specific colors, defined by [TriVertex](#) objects, on the top and bottom edges. The color SHOULD be interpolated along a gradient from the top to bottom edges and used to fill the interior.

**GRADIENT\_FILL\_TRIANGLE:** In this mode, an array of [TriVertex](#) objects is specified along with a list of array indexes that describes separate triangles. Linear interpolation SHOULD be performed between triangle vertexes and used to fill the interior. Drawing SHOULD be done directly in 24- and 32-bpp modes. **Dithering** SHOULD be performed in 4-, 8-, and 16-bpp modes.

## 2.1.16 GraphicsMode Enumeration

The Enhanced Metafile Format (EMF) **GraphicsMode** enumeration is used to specify how to interpret shape data such as rectangle coordinates.

```
typedef enum
{
    GM_COMPATIBLE = 0x00000001,
    GM_ADVANCED = 0x00000002
} GraphicsMode;
```

**GM\_COMPATIBLE:** TrueType text MUST be written from left to right and right side up, even if the rest of the graphics is flipped on the x or y axis because of the current world-to-device transformation in the playback device context. Only the height of the text SHOULD be scaled.

Arcs MUST be drawn using the current arc direction in the playback device context, but they MUST NOT respect the current world-to-device transformation that may require a flip along the x or y axis.

The world-to-device transformation SHOULD only be modified by changing the window and viewport extents and origins, using the [EMR\\_SETWINDOWEXTEX \(section 2.3.11.30\)](#) and [EMR\\_SETVIEWPORTEXTEX \(section 2.3.11.28\)](#) records, and the [EMR\\_SETWINDOWORGEX \(section 2.3.11.31\)](#) and [EMR\\_SETVIEWPORTORGEX \(section 2.3.11.30\)](#) records, respectively.

Changing the transformation directly by using the [EMR\\_MODIFYWORLDTRANSFORM \(section 2.3.12.1\)](#) or [EMR\\_SETWORLDTRANSFORM \(section 2.3.12.2\)](#) records MAY NOT [<12>](#) be supported.

In **GM\_COMPATIBLE** graphics mode, bottom and rightmost edges MUST be excluded when rectangles are drawn.

**GM\_ADVANCED:** TrueType text output MUST fully conform to the current world-to-device transformation in the playback device context.

Arcs MUST be drawn in the counterclockwise direction in world space; however, both arc control points and the arcs themselves MUST fully respect the current world-to-device transformation in the playback device context.

The world-to-device transform MAY [<13>](#) be modified directly by using the [EMR\\_MODIFYWORLDTRANSFORM](#) or [EMR\\_SETWORLDTRANSFORM](#) records, or indirectly by changing the window and viewport extents and origins, using the [EMR\\_SETWINDOWEXTEX \(section 2.3.11.30\)](#) and [EMR\\_SETVIEWPORTEXTEX \(section 2.3.11.28\)](#) records, and the [EMR\\_SETWINDOWORGEX \(section 2.3.11.31\)](#) and [EMR\\_SETVIEWPORTORGEX \(section 2.3.11.30\)](#) records, respectively.

In **GM\_ADVANCED** graphics mode, bottom and rightmost edges MUST be included when rectangles are drawn.

## 2.1.17 HatchStyle Enumeration

The Enhanced Metafile Format **HatchStyle** enumeration is an extension to the [Windows Metafile Format](#) **HatchStyle** enumeration defined in [MS-WMF] section **2.1.14**.

```
typedef enum
{
```

```

HS_SOLIDCLR = 0x0006,
HS_DITHEREDCLR = 0x0007,
HS_SOLIDTEXTCLR = 0x0008,
HS_DITHEREDTEXTCLR = 0x0009,
HS_SOLIDBKCLR = 0x000A,
HS_DITHEREDBKCLR = 0x000B
} HatchStyle;

```

**HS\_SOLIDCLR:** The hatch is not a pattern, but is a solid color.

**HS\_DITHEREDCLR:** The hatch is not a pattern, but is a dithered color.

**HS\_SOLIDTEXTCLR:** The hatch is not a pattern, but is a solid color, defined by the current text (foreground) color.

**HS\_DITHEREDTEXTCLR:** The hatch is not a pattern, but is a dithered color, defined by the current text (foreground) color.

**HS\_SOLIDBKCLR:** The hatch is not a pattern, but is a solid color, defined by the current background color.

**HS\_DITHEREDBKCLR:** The hatch is not a pattern, but is a dithered color, defined by the current background color.

### 2.1.18 ICMMode Enumeration

The Enhanced Metafile Format **ICMMode** enumeration defines values that specify when to turn on and off Image Color Management (ICM).

```

typedef enum
{
    ICM_OFF = 0x01,
    ICM_ON = 0x02,
    ICM_QUERY = 0x03,
    ICM_DONE_OUTSIDEDC = 0x04
} ICMMode;

```

**ICM\_OFF:** Turns off color management. Turns on old-style **color correction** of halftones.

**ICM\_ON:** Turns on color management. Turns off old-style color correction of halftones.

**ICM\_QUERY:** Queries the current state of color management.

**ICM\_DONE\_OUTSIDEDC:** Turns off color management inside the playback device context, and turns off old-style color correction of halftones.

### 2.1.19 Illuminant Enumeration

The Enhanced Metafile Format **Illuminant** enumeration defines values that specify the illuminant value of an image, which determine the standard light source under which the image is viewed, so that the color can be adjusted appropriately.

```

typedef enum
{

```

```

    ILLUMINANT_DEVICE_DEFAULT = 0x00,
    ILLUMINANT_TUNGSTEN = 0x01,
    ILLUMINANT_B = 0x02,
    ILLUMINANT_DAYLIGHT = 0x03,
    ILLUMINANT_D50 = 0x04,
    ILLUMINANT_D55 = 0x05,
    ILLUMINANT_D65 = 0x06,
    ILLUMINANT_D75 = 0x07,
    ILLUMINANT_FLUORESCENT = 0x08
} Illuminant;

```

**ILLUMINANT\_DEVICE\_DEFAULT:** Device's default. Standard used by output devices.

**ILLUMINANT\_TUNGSTEN:** Tungsten lamp.

**ILLUMINANT\_B:** Noon sunlight.

**ILLUMINANT\_DAYLIGHT:** Daylight.

**ILLUMINANT\_D50:** Normal print.

**ILLUMINANT\_D55:** Bond paper print.

**ILLUMINANT\_D65:** Standard daylight. Standard for CRTs and pictures.

**ILLUMINANT\_D75:** Northern daylight.

**ILLUMINANT\_FLUORESCENT:** Cool white lamp.

## 2.1.20 Letterform Enumeration

The Enhanced Metafile Format **Letterform** enumeration defines values for one of the characteristics in the Panose system for classifying typefaces.

```

typedef enum
{
    PAN_ANY = 0x00,
    PAN_NO_FIT = 0x01,
    PAN_LETT_NORMAL_CONTACT = 0x02,
    PAN_LETT_NORMAL_WEIGHTED = 0x03,
    PAN_LETT_NORMAL_BOXED = 0x04,
    PAN_LETT_NORMAL_FLATTENED = 0x05,
    PAN_LETT_NORMAL_ROUNDED = 0x06,
    PAN_LETT_NORMAL_OFF_CENTER = 0x07,
    PAN_LETT_NORMAL_SQUARE = 0x08,
    PAN_LETT_OBLIQUE_CONTACT = 0x09,
    PAN_LETT_OBLIQUE_WEIGHTED = 0x0A,
    PAN_LETT_OBLIQUE_BOXED = 0x0B,
    PAN_LETT_OBLIQUE_FLATTENED = 0x0C,
    PAN_LETT_OBLIQUE_ROUNDED = 0x0D,
    PAN_LETT_OBLIQUE_OFF_CENTER = 0x0E,
    PAN_LETT_OBLIQUE_SQUARE = 0x0F
} Letterform;

```

**PAN\_ANY:** Any.

**PAN\_NO\_FIT:** No fit.

**PAN\_LETT\_NORMAL\_CONTACT:** Normal/contact.

**PAN\_LETT\_NORMAL\_WEIGHTED:** Normal/weighted.

**PAN\_LETT\_NORMAL\_BOXED:** Normal/boxed.

**PAN\_LETT\_NORMAL\_FLATTENED:** Normal/flattened.

**PAN\_LETT\_NORMAL\_ROUNDED:** Normal/rounded.

**PAN\_LETT\_NORMAL\_OFF\_CENTER:** Normal/off center.

**PAN\_LETT\_NORMAL\_SQUARE:** Normal/square

**PAN\_LETT\_OBLIQUE\_CONTACT:** Oblique/contact.

**PAN\_LETT\_OBLIQUE\_WEIGHTED:** Oblique/weighted.

**PAN\_LETT\_OBLIQUE\_BOXED:** Oblique/boxed.

**PAN\_LETT\_OBLIQUE\_FLATTENED:** Oblique/flattened.

**PAN\_LETT\_OBLIQUE\_ROUNDED:** Oblique/rounded.

**PAN\_LETT\_OBLIQUE\_OFF\_CENTER:** Oblique/off center.

**PAN\_LETT\_OBLIQUE\_SQUARE:** Oblique/square.

### 2.1.21 MapMode Enumeration

The Enhanced Metafile Format **MapMode** enumeration is used to define the unit of measure for transforming page-space units into device-space units, and the orientation of the drawing axes.

```
typedef enum
{
    MM_TEXT = 0x01,
    MM_LOMETRIC = 0x02,
    MM_HIMETRIC = 0x03,
    MM_LOENGLISH = 0x04,
    MM_HIENGLISH = 0x05,
    MM_TWIPS = 0x06,
    MM_ISOTROPIC = 0x07,
    MM_ANISOTROPIC = 0x08
} MapMode;
```

**MM\_TEXT:** Each logical unit is mapped to one device pixel. Positive x is to the right; positive y is down.

**MM\_LOMETRIC:** Each logical unit is mapped to 0.1 millimeter. Positive x is to the right; positive y is up.

**MM\_HIMETRIC:** Each logical unit is mapped to 0.01 millimeter. Positive x is to the right; positive y is up.

**MM\_LOENGLISH:** Each logical unit is mapped to 0.01 inch. Positive x is to the right; positive y is up.

**MM\_HIENGLISH:** Each logical unit is mapped to 0.001 inch. Positive x is to the right; positive y is up.

**MM\_TWIPS:** Each logical unit is mapped to one twentieth of a printer's point (1/1440 inch, also called a twip). Positive x is to the right; positive y is up.

**MM\_ISOTROPIC:** Logical units are mapped to arbitrary units with equally-scaled axes; that is, one unit along the x-axis is equal to one unit along the y-axis. The [EMR\\_SETWINDOWEXTTEXT](#) and [EMR\\_SETVIEWPORTTEXTTEXT](#) records SHOULD be used to specify the units and the orientation of the axes.

Adjustments MUST be made as necessary to ensure the x and y units remain the same size. For example, when the window extent is set, the viewport MUST be adjusted to keep the units isotropic.

**MM\_ANISOTROPIC:** Logical units are mapped to arbitrary units with arbitrarily scaled axes. The [EMR\\_SETWINDOWEXTTEXT](#) and [EMR\\_SETVIEWPORTTEXTTEXT](#) records SHOULD be used to specify the units, orientation, and scaling.

### 2.1.22 MetafileVersion Enumeration

The Enhanced Metafile Format (EMF) **MetafileVersion** enumeration defines versions of EMF **metafiles**.

```
typedef enum
{
    META_FORMAT_WINDOWS = 0x00000300,
    META_FORMAT_ENHANCED = 0x00010000
} MetafileVersion;
```

**META\_FORMAT\_WINDOWS:** The EMF metafile MUST be interoperable with Windows 95 operating system technology, which includes Windows 95, Windows 98, and Windows Me. [<14>](#)

**META\_FORMAT\_ENHANCED:** The EMF metafile MUST be interoperable with Windows NT operating system technology, which includes Windows NT 3.1, Windows NT 3.51, Windows NT 4.0, Windows 2000, Windows Server 2003, Windows XP, Windows Vista, and Windows Server 2008. [<15>](#)

### 2.1.23 MidLine Enumeration

The Enhanced Metafile Format **MidLine** enumeration defines values for one of the characteristics in the Panose system for classifying typefaces.

```
typedef enum
{
    PAN_ANY = 0x00,
    PAN_NO_FIT = 0x01,
    PAN_MIDLINE_STANDARD_TRIMMED = 0x02,
    PAN_MIDLINE_STANDARD_POINTED = 0x03,
    PAN_MIDLINE_STANDARD_SERIFED = 0x04,
    PAN_MIDLINE_HIGH_TRIMMED = 0x05,
```

```

PAN_MIDLINE_HIGH_POINTED = 0x06,
PAN_MIDLINE_HIGH_SERIFED = 0x07,
PAN_MIDLINE_CONSTANT_TRIMMED = 0x08,
PAN_MIDLINE_CONSTANT_POINTED = 0x09,
PAN_MIDLINE_CONSTANT_SERIFED = 0x0A,
PAN_MIDLINE_LOW_TRIMMED = 0x0B,
PAN_MIDLINE_LOW_POINTED = 0x0C,
PAN_MIDLINE_LOW_SERIFED = 0x0D
} MidLine;

```

**PAN\_ANY:** Any.

**PAN\_NO\_FIT:** No fit.

**PAN\_MIDLINE\_STANDARD\_TRIMMED:** Standard/trimmed.

**PAN\_MIDLINE\_STANDARD\_POINTED:** Standard/pointed.

**PAN\_MIDLINE\_STANDARD\_SERIFED:** Standard/serifed.

**PAN\_MIDLINE\_HIGH\_TRIMMED:** High/trimmed.

**PAN\_MIDLINE\_HIGH\_POINTED:** High/pointed.

**PAN\_MIDLINE\_HIGH\_SERIFED:** High/serifed.

**PAN\_MIDLINE\_CONSTANT\_TRIMMED:** Constant/trimmed.

**PAN\_MIDLINE\_CONSTANT\_POINTED:** Constant/pointed.

**PAN\_MIDLINE\_CONSTANT\_SERIFED:** Constant/serifed.

**PAN\_MIDLINE\_LOW\_TRIMMED:** Low/trimmed.

**PAN\_MIDLINE\_LOW\_POINTED:** Low/pointed.

**PAN\_MIDLINE\_LOW\_SERIFED:** Low/serifed.

## 2.1.24 ModifyWorldTransformMode Enumeration

The Enhanced Metafile Format **ModifyWorldTransformMode** enumeration is used to specify how transformation data modifies the current world transformation.

```

typedef enum
{
    MWT_IDENTITY = 0x01,
    MWT_LEFTMULTIPLY = 0x02,
    MWT_RIGHTMULTIPLY = 0x03,
    MWT_SET = 0x04
} ModifyWorldTransformMode;

```

**MWT\_IDENTITY:** Resets the current world transformation by using the identity matrix. If this mode is specified, the [XForm](#) object is ignored.



**MWT\_LEFTMULTIPLY:** Multiplies the current transformation by the data in the XForm object. (The data in the XForm object becomes the left multiplicand, and the data for the current transformation becomes the right multiplicand.)

**MWT\_RIGHTMULTIPLY:** Multiplies the current transformation by the data in the XForm object. (The data in the XForm object becomes the right multiplicand, and the data for the current transformation becomes the left multiplicand.)

**MWT\_SET:** Same as using an [EMR\\_MODIFYWORLDTRANSFORM](#) record.

### 2.1.25 PenStyle Enumeration

The 32-bit Enhanced Metafile Format **PenStyle** enumeration is used to specify different types of pens that can be used in graphics operations.

The various pen styles specified by this enumeration can be combined, using a logical OR statement, one from each subsection of Style, EndCap, Join, and Type (Cosmetic or Geometric).

```
typedef enum
{
    PS_COSMETIC = 0x00000000,
    PS_ENDCAP_ROUND = 0x00000000,
    PS_JOIN_ROUND = 0x00000000,
    PS_SOLID = 0x00000000,
    PS_DASH = 0x00000001,
    PS_DOT = 0x00000002,
    PS_DASHDOT = 0x00000003,
    PS_DASHDOTDOT = 0x00000004,
    PS_NULL = 0x00000005,
    PS_INSIDEFRAME = 0x00000006,
    PS_USERSTYLE = 0x00000007,
    PS_ALTERNATE = 0x00000008,
    PS_ENDCAP_SQUARE = 0x00000100,
    PS_ENDCAP_FLAT = 0x00000200,
    PS_JOIN_BEVEL = 0x00001000,
    PS_JOIN_MITER = 0x00002000,
    PS_GEOMETRIC = 0x00010000
} PenStyle;
```

**PS\_COSMETIC:** The pen is cosmetic.

**PS\_ENDCAP\_ROUND:** Line end caps are round.

**PS\_JOIN\_ROUND:** Line joins are round.

**PS\_SOLID:** The pen is solid.

**PS\_DASH:** The pen is dashed.

**PS\_DOT:** The pen is dotted.

**PS\_DASHDOT:** The pen has alternating dashes and dots.

**PS\_DASHDOTDOT:** The pen has dashes and double dots.

**PS\_NULL:** The pen is invisible.

**PS\_INSIDEFRAME:** The pen is solid. When this pen is used in any drawing record that takes a bounding rectangle, the dimensions of the figure are shrunk so that it fits entirely in the bounding rectangle, taking into account the width of the pen.

**PS\_USERSTYLE:** The pen uses a styling array supplied by the user.

**PS\_ALTERNATE:** The pen sets every other pixel(this style is applicable only for cosmetic pens).

**PS\_ENDCAP\_SQUARE:** Line end caps are square.

**PS\_ENDCAP\_FLAT:** Line end caps are flat.

**PS\_JOIN\_BEVEL:** Line joins are beveled.

**PS\_JOIN\_MITER:** Line joins are mitered when they are within the current limit set by the [EMR\\_SETMITERLIMIT](#) record. A join is beveled when it would exceed the limit.

**PS\_GEOMETRIC:** The pen is geometric.

### 2.1.26 Point Enumeration

The Enhanced Metafile Format **Point** enumeration is used to specify how a point is to be used in a drawing call.

```
typedef enum
{
    PT_CLOSEFIGURE = 0x01,
    PT_LINETO = 0x02,
    PT_BEZIERTO = 0x04,
    PT_MOVETO = 0x06
} Point;
```

**PT\_CLOSEFIGURE:** A **PT\_LINETO** or **PT\_BEZIERTO** type can be combined with this value by using the bitwise operator OR to indicate that the corresponding point is the last point in a figure and the figure is closed.

The current position is set to the ending point of the closing line.

**PT\_LINETO:** Specifies that a line is to be drawn from the current position to this point, which then becomes the new current position.

**PT\_BEZIERTO:** Specifies that this point is a control point or ending point for a Bezier curve.

**PT\_BEZIERTO** types always occur in sets of three. The current position defines the starting point for the Bezier curve. The first two **PT\_BEZIERTO** points are the control points, and the third **PT\_BEZIERTO** point is the ending point. The ending point becomes the new current position. If there are not three consecutive **PT\_BEZIERTO** points, an error results.

**PT\_MOVETO:** Specifies that this point starts a disjoint figure. This point becomes the new current position.

### 2.1.27 PolygonFillMode Enumeration

The Enhanced Metafile Format **PolygonFillMode** enumeration defines values that specify how to calculate the region of a polygon is to be filled.

```
typedef enum
{
    ALTERNATE = 0x01,
    WINDING = 0x02
} PolygonFillMode;
```

**ALTERNATE:** Selects alternate mode (fills the area between odd-numbered and even-numbered polygon sides on each scan line).

**WINDING:** Selects winding mode (fills any region with a nonzero winding value).

## 2.1.28 Proportion Enumeration

The Enhanced Metafile Format **Proportion** enumeration defines values for one of the characteristics in the Panose system for classifying typefaces.

```
typedef enum
{
    PAN_ANY = 0x00,
    PAN_NO_FIT = 0x01,
    PAN_PROP_OLD_STYLE = 0x02,
    PAN_PROP_MODERN = 0x03,
    PAN_PROP_EVEN_WIDTH = 0x04,
    PAN_PROP_EXPANDED = 0x05,
    PAN_PROP_CONDENSED = 0x06,
    PAN_PROP_VERY_EXPANDED = 0x07,
    PAN_PROP_VERY_CONDENSED = 0x08,
    PAN_PROP_MONOSPACED = 0x09
} Proportion;
```

**PAN\_ANY:** Any.

**PAN\_NO\_FIT:** No fit.

**PAN\_PROP\_OLD\_STYLE:** Old style.

**PAN\_PROP\_MODERN:** Modern.

**PAN\_PROP\_EVEN\_WIDTH:** Even width.

**PAN\_PROP\_EXPANDED:** Expanded.

**PAN\_PROP\_CONDENSED:** Condensed.

**PAN\_PROP\_VERY\_EXPANDED:** Very expanded.

**PAN\_PROP\_VERY\_CONDENSED:** Very condensed.

**PAN\_PROP\_MONOSPACED:** Monospaced.

## 2.1.29 RegionMode Enumeration

The Enhanced Metafile Format **RegionMode** enumeration defines values that are used with [EMR\\_SELECTCLIPPATH](#) and [EMR\\_EXTSELECTCLIPRGN](#), specifying the current path or a new region that is being combined with the current clip region.

```
typedef enum
{
    RGN_AND = 0x01,
    RGN_OR = 0x02,
    RGN_XOR = 0x03,
    RGN_DIFF = 0x04,
    RGN_COPY = 0x05
} RegionMode;
```

**RGN\_AND:** The new clipping region includes the intersection (overlapping areas) of the current clipping region and the current path (or new region).

**RGN\_OR:** The new clipping region includes the union (combined areas) of the current clipping region and the current path (or new region).

**RGN\_XOR:** The new clipping region includes the union of the current clipping region and the current path (or new region) but without the overlapping areas.

**RGN\_DIFF:** The new clipping region includes the areas of the current clipping region with those of the current path (or new region) excluded.

**RGN\_COPY:** The new clipping region is the current path (or the new region).

### 2.1.30 SerifType Enumeration

The Enhanced Metafile Format **SerifType** enumeration defines values for one of the characteristics in the Panose system for classifying typefaces.

```
typedef enum
{
    PAN_ANY = 0x00,
    PAN_NO_FIT = 0x01,
    PAN_SERIF_COVE = 0x02,
    PAN_SERIF_OBTUSE_COVE = 0x03,
    PAN_SERIF_SQUARE_COVE = 0x04,
    PAN_SERIF_OBTUSE_SQUARE_COVE = 0x05,
    PAN_SERIF_SQUARE = 0x06,
    PAN_SERIF_THIN = 0x07,
    PAN_SERIF_BONE = 0x08,
    PAN_SERIF_EXAGGERATED = 0x09,
    PAN_SERIF_TRIANGLE = 0x0A,
    PAN_SERIF_NORMAL_SANS = 0x0B,
    PAN_SERIF_OBTUSE_SANS = 0x0C,
    PAN_SERIF_PERP_SANS = 0x0D,
    PAN_SERIF_FLARED = 0x0E,
    PAN_SERIF_ROUNDED = 0x0F
} SerifType;
```

**PAN\_ANY:** Any.

**PAN\_NO\_FIT:** No fit.

**PAN\_SERIF\_COVE:** Cove.

**PAN\_SERIF\_OBTUSE\_COVE:** Obtuse cove.

**PAN\_SERIF\_SQUARE\_COVE:** Square cove.

**PAN\_SERIF\_OBTUSE\_SQUARE\_COVE:** Obtuse square cove.

**PAN\_SERIF\_SQUARE:** Square.

**PAN\_SERIF\_THIN:** Thin.

**PAN\_SERIF\_BONE:** Bone.

**PAN\_SERIF\_EXAGGERATED:** Exaggerated.

**PAN\_SERIF\_TRIANGLE:** Triangle.

**PAN\_SERIF\_NORMAL\_SANS:** Normal sans.

**PAN\_SERIF\_OBTUSE\_SANS:** Obtuse sans.

**PAN\_SERIF\_PERP\_SANS:** Perp sans.

**PAN\_SERIF\_FLARED:** Flared.

**PAN\_SERIF\_ROUNDED:** Rounded.

### 2.1.31 StockObject Enumeration

The Enhanced Metafile Format (EMF) **StockObject** enumeration specifies a set of pre-defined graphics objects that can be selected into the playback device context in the course of metafile processing.

```
typedef enum
{
    WHITE_BRUSH = 0x80000000,
    LTGRAY_BRUSH = 0x80000001,
    GRAY_BRUSH = 0x80000002,
    DKGRAY_BRUSH = 0x80000003,
    BLACK_BRUSH = 0x80000004,
    NULL_BRUSH = 0x80000005,
    WHITE_PEN = 0x80000006,
    BLACK_PEN = 0x80000007,
    NULL_PEN = 0x80000008,
    OEM_FIXED_FONT = 0x8000000A,
    ANSI_FIXED_FONT = 0x8000000B,
    ANSI_VAR_FONT = 0x8000000C,
    SYSTEM_FONT = 0x8000000D,
    DEVICE_DEFAULT_FONT = 0x8000000E,
    DEFAULT_PALETTE = 0x8000000F,
    SYSTEM_FIXED_FONT = 0x80000010,
    DEFAULT_GUI_FONT = 0x80000011,
    DC_BRUSH = 0x80000012,
    DC_PEN = 0x80000013
} StockObject;
```

**WHITE\_BRUSH:** White solid-color brush.

**LTGRAY\_BRUSH:** Light gray solid-color or dithered brush.

**GRAY\_BRUSH:** Gray solid-color or dithered brush.

**DKGRAY\_BRUSH:** Dark gray solid color or dithered brush.

**BLACK\_BRUSH:** Black solid color brush.

**NULL\_BRUSH:** Null brush. [<16>](#)

**WHITE\_PEN:** White pen.

**BLACK\_PEN:** Black pen.

**NULL\_PEN:** Null pen. A null pen draws nothing.

**OEM\_FIXED\_FONT:** A fixed-pitch font using an **Original Equipment Manufacturer (OEM) character set** encoding.

**ANSI\_FIXED\_FONT:** A fixed-pitch font using the **ANSI character set** encoding. [<17>](#)

**ANSI\_VAR\_FONT:** A variable-pitch font using the ANSI character set encoding. [<18>](#)

**SYSTEM\_FONT:** A variable-pitch font using the ANSI character set encoding, which is guaranteed to be available in the operating system. [<19>](#)

**DEVICE\_DEFAULT\_FONT:** [<20>](#)

**DEFAULT\_PALETTE:** Default palette. This palette consists of the static colors in the system palette.

**SYSTEM\_FIXED\_FONT:** A fixed-pitch font using the ANSI character set encoding, which is guaranteed to be available in the operating system.

**DEFAULT\_GUI\_FONT:** The default system font for user interface objects such as menus and dialog boxes.

**DC\_BRUSH:** The solid-color system brush, which is currently selected in the playback device context. [<21>](#)

**DC\_PEN:** The solid-color system pen, which is currently selected in the playback device context. [<22>](#)

Unlike graphics objects, stock objects are not explicitly created, nor can they be deleted; their indexes SHOULD be used only in the [EMR\\_SELECTOBJECT \(section 2.3.8.5\)](#) record. The index of a stock object is distinguishable from the index of a dynamic graphics object by the setting of the most-significant bit. If the bit is set, the object is a stock object; if the bit is clear, it is a graphics object that was created by a previous record in the metafile.

### 2.1.32 StretchMode Enumeration

The Enhanced Metafile Format **StretchMode** enumeration is used to specify how color data is added to or removed from bitmaps that are stretched or compressed. [<23>](#)

```
typedef enum
{
    STRETCH_ANDSCANS = 0x01,
    STRETCH_ORSCANS = 0x02,
    STRETCH_DELETESCANS = 0x03,
    STRETCH_HALFTONE = 0x04
}
```

```
} StretchMode;
```

**STRETCH\_ANDSCANS:** Performs a Boolean AND operation using the color values for the eliminated and existing pixels. If the bitmap is a monochrome bitmap, this mode preserves black pixels at the expense of white pixels.

**STRETCH\_ORSCANS:** Performs a Boolean OR operation using the color values for the eliminated and existing pixels. If the bitmap is a monochrome bitmap, this mode preserves white pixels at the expense of black pixels.

**STRETCH\_DELETESCANS:** Deletes the pixels. This mode deletes all eliminated lines of pixels without trying to preserve their information.

**STRETCH\_HALFTONE:** Maps pixels from the source rectangle into blocks of pixels in the destination rectangle. The average color over the destination block of pixels approximates the color of the source pixels.

After setting the **STRETCH\_HALFTONE** stretching mode, the brush origin SHOULD be defined by an [EMR\\_SETBRUSHORGE](#) record. If it fails to do so, brush misalignment occurs.

### 2.1.33 StrokeVariation Enumeration

The Enhanced Metafile Format **StrokeVariation** enumeration defines values for one of the characteristics in the Panose system for classifying typefaces.

```
typedef enum
{
    PAN_ANY = 0x00,
    PAN_NO_FIT = 0x01,
    PAN_STROKE_GRADUAL_DIAG = 0x02,
    PAN_STROKE_GRADUAL_TRAN = 0x03,
    PAN_STROKE_GRADUAL_VERT = 0x04,
    PAN_STROKE_GRADUAL_HORZ = 0x05,
    PAN_STROKE_RAPID_VERT = 0x06,
    PAN_STROKE_RAPID_HORZ = 0x07,
    PAN_STROKE_INSTANT_VERT = 0x08
} StrokeVariation;
```

**PAN\_ANY:** Any.

**PAN\_NO\_FIT:** No fit.

**PAN\_STROKE\_GRADUAL\_DIAG:** Gradual/diagonal.

**PAN\_STROKE\_GRADUAL\_TRAN:** Gradual/transitional.

**PAN\_STROKE\_GRADUAL\_VERT:** Gradual/vertical.

**PAN\_STROKE\_GRADUAL\_HORZ:** Gradual/horizontal.

**PAN\_STROKE\_RAPID\_VERT:** Rapid/vertical.

**PAN\_STROKE\_RAPID\_HORZ:** Rapid/horizontal.

**PAN\_STROKE\_INSTANT\_VERT:** Instant/vertical.

### 2.1.34 TextAlignmentMode Enumeration

The Enhanced Metafile Format (EMF) **TextAlignmentMode** enumeration is used to specify the relationship between a reference point and a rectangle that bounds text. This relationship **MUST** be considered when positioning a string of text.

```
typedef enum
{
    TA_LEFT = 0x0000,
    TA_NOUPDATECP = 0x0000,
    TA_TOP = 0x0000,
    TA_UPDATECP = 0x0001,
    TA_RIGHT = 0x0002,
    TA_CENTER = 0x0006,
    TA_BOTTOM = 0x0008,
    TA_BASELINE = 0x0018,
    TA_RTLREADING = 0x0100
} TextAlignmentMode;
```

**TA\_LEFT:** The reference point will be on the left edge of the bounding rectangle.

**TA\_NOUPDATECP:** The current position is not updated after each text output call. The reference point is specified in the text output record.

**TA\_TOP:** The reference point will be on the top edge of the bounding rectangle.

**TA\_UPDATECP:** The current position is updated after each text output call. The current position is used as the reference point.

**TA\_RIGHT:** The reference point will be on the right edge of the bounding rectangle.

**TA\_CENTER:** The reference point will be aligned horizontally with the center of the bounding rectangle.

**TA\_BOTTOM:** The reference point will be on the bottom edge of the bounding rectangle.

**TA\_BASELINE:** The reference point will be on the base line of the text.

**TA\_RTLREADING:** The text is laid out in right to left reading order, as opposed to the default left to right order.

### 2.1.35 VerticalTextAlignmentMode Enumeration

The Enhanced Metafile Format **VerticalTextAlignmentMode** enumeration is used instead of [TextAlignmentMode](#), when the font has a vertical default baseline, as with Kanji.

```
typedef enum
{
    VTA_CENTER = 0x0006,
    VTA_BASELINE = 0x0018
} VerticalTextAlignmentMode;
```

**VTA\_CENTER:** The reference point will be aligned vertically with the center of the bounding rectangle.



**VTA\_BASELINE:** The reference point will be on the base line of the text.

### 2.1.36 Weight Enumeration

The Enhanced Metafile Format **Weight** enumeration defines values for one of the characteristics in the Panose system for classifying typefaces.

```
typedef enum
{
    PAN_ANY = 0x00,
    PAN_NO_FIT = 0x01,
    PAN_WEIGHT_VERY_LIGHT = 0x02,
    PAN_WEIGHT_LIGHT = 0x03,
    PAN_WEIGHT_THIN = 0x04,
    PAN_WEIGHT_BOOK = 0x05,
    PAN_WEIGHT_MEDIUM = 0x06,
    PAN_WEIGHT_DEMI = 0x07,
    PAN_WEIGHT_BOLD = 0x08,
    PAN_WEIGHT_HEAVY = 0x09,
    PAN_WEIGHT_BLACK = 0x0A,
    PAN_WEIGHT_NORD = 0x0B
} Weight;
```

**PAN\_ANY:** Any.

**PAN\_NO\_FIT:** No fit.

**PAN\_WEIGHT\_VERY\_LIGHT:** Very light.

**PAN\_WEIGHT\_LIGHT:** Light.

**PAN\_WEIGHT\_THIN:** Thin.

**PAN\_WEIGHT\_BOOK:** Book.

**PAN\_WEIGHT\_MEDIUM:** Medium.

**PAN\_WEIGHT\_DEMI:** Demi.

**PAN\_WEIGHT\_BOLD:** Bold.

**PAN\_WEIGHT\_HEAVY:** Heavy.

**PAN\_WEIGHT\_BLACK:** Black.

**PAN\_WEIGHT\_NORD:** Nord.

### 2.1.37 XHeight Enumeration

The Enhanced Metafile Format **XHeight** enumeration defines values for one of the characteristics in the Panose system for classifying typefaces.

```
typedef enum
{
    PAN_ANY = 0x00,
    PAN_NO_FIT = 0x01,
```

```

PAN_XHEIGHT_CONSTANT_SMALL = 0x02,
PAN_XHEIGHT_CONSTANT_STD = 0x03,
PAN_XHEIGHT_CONSTANT_LARGE = 0x04,
PAN_XHEIGHT_DUCKING_SMALL = 0x05,
PAN_XHEIGHT_DUCKING_STD = 0x06,
PAN_XHEIGHT_DUCKING_LARGE = 0x07
} XHeight;

```

**PAN\_ANY:** Any.

**PAN\_NO\_FIT:** No fit.

**PAN\_XHEIGHT\_CONSTANT\_SMALL:** Constant/small.

**PAN\_XHEIGHT\_CONSTANT\_STD:** Constant/standard.

**PAN\_XHEIGHT\_CONSTANT\_LARGE:** Constant/large.

**PAN\_XHEIGHT\_DUCKING\_SMALL:** Ducking/small

**PAN\_XHEIGHT\_DUCKING\_STD:** Ducking/standard.

**PAN\_XHEIGHT\_DUCKING\_LARGE:** Ducking/large.

## 2.2 EMF Objects

### 2.2.1 ColorAdjustment Object

The Enhanced Metafile Format (EMF) ColorAdjustment object defines color adjustment values used by the [EMR\\_STRETCHBLT](#) and [EMR\\_STRETCHDIBITS](#) records when the [StretchMode \(section 2.1.32\)](#) is **STRETCH\_HALFTONE**.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Size																Values															
IlluminantIndex																RedGamma															
GreenGamma																BlueGamma															
ReferenceBlack																ReferenceWhite															
Contrast																Brightness															
Colorfulness																RedGreenTint															

**Size (2 bytes):** A 16-bit unsigned integer that specifies the size, in bytes, of this object.

**Values (2 bytes):** A 16-bit unsigned integer that specifies how the output image SHOULD be prepared. This field MAY be set to NULL or any combination of values in the **ColorAdjustment** enumeration table, as specified in section [2.1.5](#).

**IlluminantIndex (2 bytes):** A 16-bit unsigned integer that specifies the type of standard light source under which the image is viewed. The value MUST be in the **Illuminant** enumeration table, as specified in section [2.1.19](#).

**RedGamma (2 bytes):** A 16-bit unsigned integer that specifies the nth power **gamma correction** value for the red primary of the source colors. The value MUST be in the range from 2,500 to 65,000. A value of 10,000 means gamma correction MUST NOT be performed.

**GreenGamma (2 bytes):** A 16-bit unsigned integer that specifies the nth power gamma correction value for the green primary of the source colors. The value MUST be in the range from 2500 to 65,000. A value of 10,000 means gamma correction MUST NOT be performed.

**BlueGamma (2 bytes):** A 16-bit unsigned integer that specifies the nth power gamma correction value for the blue primary of the source colors. The value MUST be in the range from 2500 to 65,000. A value of 10,000 means gamma correction MUST NOT be performed.

**ReferenceBlack (2 bytes):** A 16-bit unsigned integer that specifies the black reference for the source colors. Any colors that are darker than this are treated as black. The value MUST be in the range from 0 to 4000.

**ReferenceWhite (2 bytes):** A 16-bit unsigned integer that specifies the white reference for the source colors. Any colors that are lighter than this are treated as white. The value MUST be in the range from 6000 to 10,000.

**Contrast (2 bytes):** A 16-bit signed integer that specifies the amount of contrast to be applied to the source object. The value MUST be in the range from -100 to 100. A value of 0 means contrast adjustment MUST NOT be performed.

**Brightness (2 bytes):** A 16-bit signed integer that specifies the amount of brightness to be applied to the source object. The value MUST be in the range from -100 to 100. A value of 0 means brightness adjustment MUST NOT be performed.

**Colorfulness (2 bytes):** A 16-bit signed integer that specifies the amount of colorfulness to be applied to the source object. The value MUST be in the range from -100 to 100. A value of 0 means colorfulness adjustment MUST NOT be performed.

**RedGreenTint (2 bytes):** A 16-bit signed integer that specifies the amount of red or green tint adjustment to be applied to the source object. The value MUST be in the range from -100 to 100. Positive numbers adjust towards red and negative numbers adjust towards green. Zero means tint adjustment MUST NOT be performed.

### 2.2.2 DesignVector Object

The Enhanced Metafile Format DesignVector object defines the **design vector**, which specifies values for the **font axes** of a **multiple master** font.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Signature																															
NumAxes																															
Values (variable)																															
...																															

**Signature (4 bytes):** A 32-bit unsigned integer that MUST be set to the value 0x80007664.

**NumAxes (4 bytes):** A 32-bit unsigned integer that specifies the number of elements in the **Values** array. It MUST be in the range 0 to 16, inclusive.

**Values (variable):** An optional array of 32-bit signed integers that specify the values of the font axes of a multiple master, **OpenType** font. The maximum number of values in the array is 16.

### 2.2.3 EmrFormat Object

The Enhanced Metafile Format EmrFormat object contains information that identifies graphics data in an EMF metafile. An [EMR\\_COMMENT\\_MULTIFORMATS \(section 2.3.3.4.3\)](#) public comment record contains an array of EmrFormat objects.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Signature																															
Version																															
SizeData																															
offData																															

**Signature (4 bytes):** A 32-bit unsigned integer that specifies the format signature. This value MUST be in the [FormatSignature \(section 2.1.14\)](#) enumeration.

**Version (4 bytes):** A 32-bit unsigned integer that specifies the format version number.

**SizeData (4 bytes):** A 32-bit unsigned integer that specifies the size of the data in bytes.

**offData (4 bytes):** A 32-bit unsigned integer that specifies the offset to the data from the start of the **EMR\_COMMENT\_PUBLIC** identifier field, defined in section [2.3.3.4](#). The offset MUST be 32-bit aligned.

## 2.2.4 EmrText Object

The Enhanced Metafile Format (EMF) EmrText object contains the fields for text output.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Reference																															
...																															
Chars																															
offString																															
Options																															
Rectangle																															
...																															
...																															
...																															
offDx																															
text (variable)																															
...																															
spacing (variable)																															
...																															

**Reference (8 bytes):** A 64-bit [Windows Metafile Format \(WMF\) PointL](#) object, as specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies the coordinates of the reference point used to position the string.

**Chars (4 bytes):** A 32-bit unsigned integer that specifies the number of characters in the string.

**offString (4 bytes):** A 32-bit unsigned integer that specifies the offset to the output string, in bytes from the start of the record in which this object is contained. This value **MUST** be 8 or 16-bit aligned, according to the character format.

**Options (4 bytes):** A 32-bit unsigned integer that specifies how to use the rectangle values specified in the **Rectangle** field. This field MAY be a combination of [ExtTextOutOptions \(section 2.1.11\)](#) enumeration values.

**Rectangle (16 bytes):** A 128-bit **WMF RectL** object, as specified in [\[MS-WMF\]](#) section 2.2.1.14, that defines the optional clipping and/or opaquing rectangle, in logical units.

**offDx (4 bytes):** A 32-bit unsigned integer that specifies the offset to an intercharacter spacing array, in bytes from the start of the record in which this object is contained. This value MUST be 32-bit aligned.

**text (variable):** The optional and variable-size character string buffer.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
UndefinedSpace1 (variable)																															
...																															
OutputString (variable)																															
...																															

**UndefinedSpace1 (variable):** An optional number of unused bytes. The **OutputString** field is not required to follow immediately the preceding portion of this structure.

**OutputString (variable):** The string to output, composed of either 8 or 16-bit characters, depending on the containing structure. Its location, in bytes offset from the start of the record in which this object is contained, is specified by the **offString** field, and its length is specified by the **Chars** field.

**spacing (variable):** The optional and variable-size character spacing buffer.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
UndefinedSpace2 (variable)																															
...																															
OutputDx (variable)																															
...																															

**UndefinedSpace2 (variable):** An optional number of unused bytes. The **OutputDx** field is not required to follow immediately the preceding portion of this structure.

**OutputDx (variable):** An array of 32-bit integers that specify the spacing between characters, in logical units. Its location, in bytes offset from the start of the record in which this object is contained, is specified by the **offDx** field, and its length is specified by the **Chars** field.

## 2.2.5 EpsData Object

The Enhanced Metafile Format EpsData object is used to contain an embedded PostScript file in the metafile.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SizeData																															
Version																															
Points (variable)																															
...																															
EPSData (variable)																															
...																															

**SizeData (4 bytes):** A 32-bit unsigned integer that specifies the size of the EpsData object in bytes.

**Version (4 bytes):** A 32-bit unsigned integer that specifies the language level (for example, 1 for level 1 PostScript).

**Points (variable):** An array of three [Windows Metafile Format PointL](#) objects, as specified in [\[MS-WMF\]](#) section 2.2.1.11, that define the output parallelogram in **28.4 FIX** device coordinates.

**EPSData (variable):** A variable length array of bytes of EPS data.

## 2.2.6 GradientRectangle Object

The Enhanced Metafile Format GradientRectangle object defines a rectangle in an [EMR\\_GRADIENTFILL](#) record, as specified in section [2.3.5.12](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
UpperLeft																															
LowerRight																															

**UpperLeft (4 bytes):** An index into an array of TriVertex objects, which specifies the upper-left vertex of a rectangle. The index **MUST** be smaller than the size of the array, as defined by the **nVer** field of the EMR\_GRADIENTFILL record.

**LowerRight (4 bytes):** An index into an array of TriVertex objects, which specifies the lower-right vertex of a rectangle. The index **MUST** be smaller than the size of the array, as defined by the **nVer** field of the EMR\_GRADIENTFILL record.

## 2.2.7 GradientTriangle Object

The Enhanced Metafile Format GradientTriangle object defines a triangle in an [EMR\\_GRADIENTFILL](#) record, as specified in section [2.3.5.12](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Vertex1																															
Vertex2																															
Vertex3																															

**Vertex1 (4 bytes):** An index into an array of TriVertex objects, which specifies a vertex of a triangle. The index **MUST** be smaller than the size of the array, as defined by the **nVer** field of the EMR\_GRADIENTFILL record.

**Vertex2 (4 bytes):** An index into an array of TriVertex objects, which specifies a vertex of a triangle. The index **MUST** be smaller than the size of the array, as defined by the **nVer** field of the EMR\_GRADIENTFILL record.

**Vertex3 (4 bytes):** An index into an array of TriVertex objects, which specifies a vertex of a triangle. The index **MUST** be smaller than the size of the array, as defined by the **nVer** field of the EMR\_GRADIENTFILL record.

## 2.2.8 HeaderExtension1 Object

The Enhanced Metafile Format HeaderExtension1 object defines the first extension to the EMF metafile header. It is a group of fields that added support for a [PixelFormatDescriptor](#) and OpenGL records. [PixelFormatDescriptor](#) is specified in section [2.2.20](#). OpenGL is specified in [\[OPENGL\]](#).



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cbPixelFormat																															
offPixelFormat																															
bOpenGL																															

**cbPixelFormat (4 bytes):** A 32-bit unsigned integer that specifies the size of PixelFormatDescriptor information. This MUST be 0x00000000 if no pixel format is set.

**offPixelFormat (4 bytes):** A 32-bit unsigned integer that specifies the offset to PixelFormatDescriptor. This MUST be 0x00000000 if no pixel format is set.

**bOpenGL (4 bytes):** A 32-bit unsigned integer that indicates whether OpenGL commands are present in the metafile.

Value	Meaning
0x00000001	OpenGL records are present in the metafile.
0x00000000	OpenGL records are not present in the metafile.

## 2.2.9 HeaderExtension2 Object

The Enhanced Metafile Format HeaderExtension2 object adds the ability to measure device surfaces in micrometers, which enhances the resolution and scalability of EMF metafiles.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MicrometersX																															
MicrometersY																															

**MicrometersX (4 bytes):** The 32-bit horizontal size of the display device for which the metafile image was generated, in micrometers.

**MicrometersY (4 bytes):** The 32-bit vertical size of the display device for which the metafile image was generated, in micrometers.

## 2.2.10 LogBrushEx Object

The Enhanced Metafile Format (EMF) LogBrushEx object defines the style, color, and pattern of a device-independent brush.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BrushStyle																															
Color																															
BrushHatch																															

**BrushStyle (4 bytes):** A 32-bit unsigned integer that specifies the brush style. The value MUST be an enumeration from Windows Metafile Format (WMF) **BrushStyle** enumeration table, as specified in [\[MS-WMF\]](#) section 2.1.4. The style values that are supported in this structure are listed later in this section. The **BS\_NULL** style SHOULD be used to specify a brush that has no effect. [<24>](#)

**Color (4 bytes):** A 32-bit WMF [ColorRef](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.7, that specifies a color. The interpretation of this field depends on the value of **BrushStyle**, as explained below.

**BrushHatch (4 bytes):** A 32-bit unsigned field that contains the brush hatch data. Its interpretation depends on the value of **BrushStyle**, as explained below.

The following table shows the relationship between the **BrushStyle**, **Color** and **BrushHatch** fields in a LogBrushEx object. Only supported brush styles are listed.

BrushStyle	Color	BrushHatch
<b>BS_SOLID</b>	SHOULD be a WMF ColorRef object, which specifies the color of the brush.	Not used, and SHOULD be ignored.
<b>BS_NULL</b>	Not used, and SHOULD be ignored.	Not used, and SHOULD be ignored.
<b>BS_HATCHED</b>	SHOULD be a WMF ColorRef object, which specifies the foreground color of the hatch pattern.	SHOULD be a value from the EMF <a href="#">HatchStyle</a> enumeration, specified in section <a href="#">2.1.17</a> , which specifies the orientation of lines used to create the hatch.

## 2.2.11 LogFont Object

The Enhanced Metafile Format LogFont object specifies the attributes of a font.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Height																															
Width																															

Escapement			
Orientation			
Weight			
Italic	Underline	StrikeOut	CharSet
OutPrecision	ClipPrecision	Quality	PitchAndFamily
Facename			
...			
...			
...			
...			
...			
...			
...			
(Facename cont'd for 8 rows)			

**Height (4 bytes):** A 32-bit signed integer that specifies the height, in logical units, of the font's character cell or character. The character height value (also known as the em height) is the character cell height value minus the internal-leading value. The font mapper interprets the value specified in **Height** in the following manner.

Value	Meaning
0 <value	The font mapper transforms this value into device units and matches it against the cell height of the available fonts.
0	The font mapper uses a default height value when it searches for a match.
value < 0	The font mapper transforms this value into device units and matches its absolute value against the character height of the available fonts.

For all height comparisons, the font mapper looks for the largest font that does not exceed the requested size.

**Width (4 bytes):** A 32-bit signed integer that specifies the average width, in logical units, of characters in the font. If **Width** is zero, an appropriate value MAY be calculated from other LogFont values, in order to find a font that has the typographer's intended aspect ratio. [<25>](#)

**Escapement (4 bytes):** A 32-bit signed integer that specifies the angle, in tenths of degrees, between the escapement vector and the x-axis of the device. The escapement vector is parallel to the base line of a row of text.

When the graphics mode is set to **GM\_ADVANCED**, the escapement angle of the string can be specified independently of the orientation angle of the string's characters. Graphics modes are specified in section [2.1.16](#)

**Orientation (4 bytes):** A 32-bit signed integer that specifies the angle, in tenths of degrees, between each character's base line and the x-axis of the device.

**Weight (4 bytes):** A 32-bit signed integer that specifies the **weight** of the font in the range 0 through 1000. For example, 400 is normal and 700 is bold. If this value is 0x00000000, a default weight MAY [<26>](#) be used.

**Italic (1 byte):** An 8-bit unsigned integer that specifies an italic font if set to 0x01; otherwise, it must be 0x00.

**Underline (1 byte):** An 8-bit unsigned integer that specifies an underlined font if set to 0x01; otherwise, it must be 0x00.

**StrikeOut (1 byte):** An 8-bit unsigned integer that specifies a strikeout font if set to 0x01; otherwise, it must be 0x00.

**CharSet (1 byte):** An 8-bit unsigned integer that specifies the character set. It MAY be set to a value in the Windows Metafile Format **CharacterSet** enumeration table, specified in [\[MS-WMF\]](#) section **2.1.5**.

**DEFAULT\_CHARSET** is set to a value based on the current system locale. For example, when the system locale is English (United States), it is set as **ANSI\_CHARSET**.

Fonts with other character sets may exist in the operating system. If an application uses a font with an unknown character set, it SHOULD NOT attempt to translate or interpret strings that are rendered with that font.

This field is important in the font mapping process. To ensure consistent results, a specific character set SHOULD be specified. If a typeface name is specified in the **Facename** field, make sure that the **CharSet** value matches the character set of the typeface.

**OutPrecision (1 byte):** A 8-bit unsigned integer that specifies the output precision. The output precision defines how closely the output MUST match the requested font's height, width, character orientation, escapement, pitch, and font type. It MAY be one of the values from the WMF **OutPrecision** enumeration table, as specified in [\[MS-WMF\]](#) section **2.1.23**.

Applications can use the **OUT\_DEVICE\_PRECIS**, **OUT\_RASTER\_PRECIS**, **OUT\_TT\_PRECIS**, and **OUT\_PS\_ONLY\_PRECIS** values to control how the font mapper chooses a font when the operating system contains more than one font with a specified name. For example, if an operating system contains a font named **Symbol** in raster and TrueType form, specifying **OUT\_TT\_PRECIS** forces the font mapper to choose the TrueType version. Specifying **OUT\_TT\_ONLY\_PRECIS** forces the font mapper to choose a TrueType font, even if it substitutes a TrueType font of another name.

**ClipPrecision (1 byte):** A 8-bit unsigned integer that specifies the clipping precision. The clipping precision defines how to clip characters that are partially outside the clipping region. It can be one or more of the values in the WMF **ClipPrecision** enumeration table, as specified in [\[MS-WMF\]](#) section **2.1.6**.

**Quality (1 byte):** A 8-bit unsigned integer that specifies the output quality. The output quality defines how carefully to attempt to match the logical-font attributes to those of an actual physical font. It **MUST** be one of the values in the WMF **FontQuality** enumeration table, as specified in [\[MS-WMF\]](#) section **2.1.12**.

**PitchAndFamily (1 byte):** A 8-bit unsigned integer that specifies the pitch and family of the font. The two low-order bits define the pitch of the font and **MUST** be one of the values in the WMF **PitchFont** enumeration table, specified in [\[MS-WMF\]](#) section **2.1.26**. Bits 4 through 7 define the font family and **MUST** be one of the values in the WMF **FamilyFont** enumeration table, as specified in [\[MS-WMF\]](#) section **2.1.10**. Font families describe the look of a font in a general way. They are intended for specifying fonts when the exact typeface desired is not available.

**Facename (64 bytes):** A string of no more than 32 Unicode characters that specifies the typeface name of the font. If the length of this string is less than 32 characters, a terminating NULL **MUST** be present, after which the remainder of this field **MUST** be ignored.

**2.2.12 LogFontEx Object**

The Enhanced Metafile Format LogFontEx object defines the attributes of an extended logical font.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LogFont																															
...																															
...																															
...																															
...																															
...																															
...																															
(LogFont cont'd for 15 rows)																															
FullName																															

...
...
...
...
...
...
...
...
(FullName cont'd for 24 rows)
Style
...
...
...
...
...
...
...
...
...
(Style cont'd for 8 rows)
Script
...
...
...

...
...
...
...
(Script cont'd for 8 rows)

**LogFont (92 bytes):** A **LogFont** object, as specified in section [2.2.11](#), which contains basic font data.

**FullName (128 bytes):** A string of no more than 64 Unicode characters that contains the font's full name. . If the length of this string is less than 64 characters, a terminating NULL MUST be present, after which the remainder of this field MUST be ignored.

**Style (64 bytes):** A string of no more than 32 Unicode characters that defines the font's style. If the length of this string is than 32 characters, a terminating NULL MUST be present, after which the remainder of this field MUST be ignored.

**Script (64 bytes):** A string of no more than 32 Unicode characters that defines the character set of the font. If the length of this string is less than 32 characters, a terminating NULL MUST be present, after which the remainder of this field MUST be ignored.

### 2.2.13 LogFontExDv Object

The Enhanced Metafile Format LogFontExDv object contains the attributes of an extended logical font.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LogFontEx																															
...																															
...																															
...																															
...																															
...																															
...																															
(LogFontEx cont'd for 79 rows)																															
DesignVector (variable)																															
...																															

**LogFontEx (348 bytes):** A [LogFontEx](#) object that defines the logical attributes of the font.

**DesignVector (variable):** A variable-length, but no bigger than 72 bytes, DesignVector object, as specified in section [2.2.2](#). A design vector SHOULD NOT be defined unless the font is a multiple master OpenType font.

## 2.2.14 LogFontPanose Object

The Enhanced Metafile Format (EMF) LogFontPanose object defines the attributes of an extended logical font.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LogFont																															
...																															
...																															



...
...
...
...
...
(LogFont cont'd for 15 rows)
FullName
...
...
...
...
...
...
...
...
(FullName cont'd for 24 rows)
Style
...
...
...
...
...
...

...	
...	
(Style cont'd for 8 rows)	
Version	
StyleSize	
Match	
Reserved	
VendorId	
Culture	
Panose	
...	
...	Padding

**LogFont (92 bytes):** A [LogFont \(section 2.2.11\)](#) object that defines the basic font information.

**FullName (128 bytes):** A string of no more than 64 Unicode characters that defines the font's full name. If the length of this string is less than 64 characters, a terminating NULL MUST be present, after which the remainder of this field MUST be ignored.

**Style (64 bytes):** A string of no more than 32 Unicode characters that defines the font's style. . If the length of this string is less than 32 characters, a terminating NULL MUST be present, after which the remainder of this field MUST be ignored.

**Version (4 bytes):** A 32-bit unsigned integer that defines the version of the font. This SHOULD be zero. Not used.

**StyleSize (4 bytes):** A 32-bit unsigned integer that specifies the point size at which the font is hinted. If set to zero, the font is hinted at the point size corresponding to the **Height** field in **LogFont**.

**Match (4 bytes):** A 32-bit unsigned integer that is a unique identifier for an enumerated font. Not used.

**Reserved (4 bytes):** A 32-bit unsigned integer that is not used. It MUST be zero.

**VendorId (4 bytes):** An array of four bytes that MUST NOT be used.

- Culture (4 bytes):** A 32-bit unsigned integer that MUST be zero and MUST NOT be used.
- Panose (10 bytes):** An 80-bit [Panose \(section 2.2.19\)](#) object that specifies the typeface of the font. If all fields of this object are zero, it is ignored.
- Padding (2 bytes):** A field that exists only to ensure 32-bit alignment of this structure. It MUST be ignored.

### 2.2.15 LogPalette Object

The Enhanced Metafile Format LogPalette object specifies a logical\_palette containing device-independent color definitions.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Version																NumberOfEntries															
PaletteEntries (variable)																															
...																															

- Version (2 bytes):** A 16-bit unsigned integer that specifies the version number of the system. It MUST be set to 0x0300.
- NumberOfEntries (2 bytes):** A 16-bit unsigned integer that defines the number of [LogPaletteEntry](#) objects in the **PaletteEntries** field.
- PaletteEntries (variable):** Specifies an array of LogPaletteEntry objects that define the color and usage of each entry in the logical\_palette.

EMF MUST define colors as device-independent values because the metafile itself is device independent.

### 2.2.16 LogPaletteEntry Object

The Enhanced Metafile Format LogPaletteEntry object defines the values that make up a single entry in a [LogPalette \(section 2.2.15\)](#) object.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reserved								Blue								Green								Red							

- Reserved (1 byte):** An 8-bit unsigned integer that SHOULD NOT be used and SHOULD be ignored.
- Blue (1 byte):** An 8-bit unsigned integer that defines the blue intensity value for the entry in a LogPalette object.
- Green (1 byte):** An 8-bit unsigned integer that defines the green intensity value for the LogPalette entry.

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

EMF MUST define colors as device-independent values because the metafile itself is device-independent.

2.2.17 LogPen Object

The Enhanced Metafile Format LogPen object defines the style, width, and color of a logical pen.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PenStyle																															
Width																															
...																															
ColorRef																															

**PenStyle (4 bytes):** A 32-bit unsigned integer that specifies the PenStyle. The value MUST be defined from the **PenStyle** enumeration table, specified in section 2.1.25.

**Width (8 bytes):** A 64-bit [Windows Metafile Format PointL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies the width of the pen by the x value. The y value is unused.

**ColorRef (4 bytes):** A 32-bit WMF **ColorRef** object, specified in [\[MS-WMF\]](#) section 2.2.1.7, that specifies the pen color value.

2.2.18 LogPenEx Object

The Enhanced Metafile Format (EMF) LogPenEx object specifies the style, width, and color of an extended logical pen.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PenStyle																															
Width																															
BrushStyle																															
ColorRef																															
BrushHatch																															
NumStyleEntries																															
StyleEntry (variable)																															
...																															

**PenStyle (4 bytes):** A 32-bit unsigned integer that specifies the pen style. The value MUST be defined from the [PenStyle](#) enumeration, specified in section [2.1.25](#).

This value defines a combination of pen attributes, including the pen type, line style, end-cap style and end-join style.

**Width (4 bytes):** A 32-bit unsigned integer that specifies the width of the line drawn by the pen. If **PS\_GEOMETRIC** is set in the **PenStyle**, field, this value is the width in logical units. Otherwise, the width is specified in device units.

**BrushStyle (4 bytes):** A 32-bit unsigned integer that specifies a brush style for the pen from the Windows Metafile Format (WMF) **BrushStyle** enumeration, specified in [\[MS-WMF\]](#) section **2.1.4**.

If **PS\_GEOMETRIC** is not set in the **PenStyle** field, the **BrushStyle** MUST be set to either **BS\_SOLID** or **BS\_HATCHED**. The value of this field MAY be **BS\_NULL**, but only if the **PS\_NULL** flag is set in the **PenStyle**. The **BS\_NULL** style SHOULD be used to specify a brush that has no effect. [<27>](#)

**ColorRef (4 bytes):** A 32-bit WMF **ColorRef** object, specified in [\[MS-WMF\]](#) section 2.2.1.7. The interpretation of this field depends on the **BrushStyle** value, as explained below.

**BrushHatch (4 bytes):** The brush hatch pattern. The definition of this field depends on the **BrushStyle** value, as explained below.

**NumStyleEntries (4 bytes):** The number of elements in the array specified in the **StyleEntry** field. This value SHOULD be zero if **PenStyle** does not specify **PS\_USERSTYLE**.

**StyleEntry (variable):** A variable-length field that defines the line style for the pen. It is an array with a finite length, but it is used as if it repeated indefinitely. The first entry in the array

specifies the length of the first dash. The second entry specifies the length of the first gap. Thereafter, lengths of dashes and gaps alternate.

The LogPenEx object includes the specification of brush attributes, so it can be used to draw lines that consist of custom or pre-defined patterns. The following table shows the relationship between the **BrushStyle**, **ColorRef** and **BrushHatch** fields in a LogPenEx object. Only supported brush styles are listed.

<b>BrushStyle</b>	<b>ColorRef</b>	<b>BrushHatch</b>
<b>BS_SOLID</b>	SHOULD be a WMF ColorRef object, specified in <a href="#">[MS-WMF]</a> section 2.2.1.7, which specifies the color of lines drawn by the pen.	Not used, and SHOULD be ignored.
<b>BS_NULL</b>	Not used, and SHOULD be ignored.	Not used, and SHOULD be ignored.
<b>BS_HATCHED</b>	SHOULD be a WMF ColorRef object, which specifies the foreground color of the hatch pattern.	SHOULD be a value from the EMF <a href="#">HatchStyle</a> enumeration that specifies the orientation of lines used to create the hatch. If <b>PS_GEOMETRIC</b> is not set in the <b>PenStyle</b> field, this field MUST be either <b>HS_SOLIDTEXTCLR (0x0008)</b> or <b>HS_SOLIDBKCLR (0x000A)</b> .
<b>BS_PATTERN</b>	The low-order word SHOULD be a value from the WMF <a href="#">ColorUsage</a> enumeration, specified in <a href="#">[MS-WMF]</a> section 2.1.7.	Not used, and SHOULD be ignored. The brush pattern is specified by a packed <a href="#">DeviceIndependentBitmap (DIB)</a> , as specified in <a href="#">[MS-WMF]</a> section 2.2.2.3.
<b>BS_DIBPATTERN</b>	The low-order word SHOULD be a value from the WMF <b>ColorUsage</b> enumeration.	Not used, and SHOULD be ignored. The brush pattern is specified by a <b>packed DIB</b> .
<b>BS_DIBPATTERNPT</b>	The low-order word SHOULD be a value from the WMF <b>ColorUsage</b> enumeration.	Not used, and SHOULD be ignored. The brush pattern is specified by a packed DIB.

## 2.2.19 Panose Object

The Enhanced Metafile Format **Panose** object describes the Panose font-classification values for a TrueType font. These characteristics are used to associate the font with other fonts of similar appearance but different names.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1				
FamilyType									SerifStyle									Weight									Proportion								
Contrast									StrokeVariation									ArmStyle									Letterform								
Midline									XHeight																										

**FamilyType (1 byte):** An 8-bit unsigned integer that specifies the family type. The value MUST be in the [FamilyType \(section 2.1.12\)](#) enumeration table.

**SerifStyle (1 byte):** An 8-bit unsigned integer that specifies the serif style. The value MUST be in the [SerifType \(section 2.1.30\)](#) enumeration table.

**Weight (1 byte):** An 8-bit unsigned integer that specifies the weight of the font. The value MUST be in the [Weight \(section 2.1.36\)](#) enumeration table.

**Proportion (1 byte):** An 8-bit unsigned integer that specifies the proportion of the font. The value MUST be in the [Proportion \(section 2.1.28\)](#) enumeration table.

**Contrast (1 byte):** An 8-bit unsigned integer that specifies the contrast of the font. The value MUST be in the [Contrast \(section 2.1.8\)](#) enumeration table.

**StrokeVariation (1 byte):** An 8-bit unsigned integer that specifies the stroke variation for the font. The value MUST be in the [StrokeVariation \(section 2.1.33\)](#) enumeration table.

**ArmStyle (1 byte):** An 8-bit unsigned integer that specifies the arm style of the font. The value MUST be in the [ArmStyle \(section 2.1.3\)](#) enumeration table.

**Letterform (1 byte):** An 8-bit unsigned integer that specifies the letterform of the font. The value MUST be in the [Letterform \(section 2.1.20\)](#) enumeration table.

**Midline (1 byte):** An 8-bit unsigned integer that specifies the midline of the font. The value MUST be in the [MidLine \(section 2.1.23\)](#) enumeration table.

**XHeight (1 byte):** An 8-bit unsigned integer that specifies the x height of the font. The value MUST be in the [XHeight \(section 2.1.37\)](#) enumeration table.

## 2.2.20 PixelFormatDescriptor Object

The Enhanced Metafile Format PixelFormatDescriptor object describes the pixel format of a drawing surface.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
nSize																nVersion															
dwFlags																															
iPixelFormat								cColorBits								cRedBits								cRedShift							
cGreenBits								cGreenShift								cBlueBits								cBlueShift							
cAlphaBits								cAlphaShift								cAccumBits								cAccumRedBits							
cAccumGreenBits								cAccumBlueBits								cAccumAlphaBits								cDepthBits							
cStencilBits								cAuxBuffers								iLayerType								bReserved							
dwLayerMask																															
dwVisibleMask																															
dwDamageMask																															

**nSize (2 bytes):** A 16-bit integer that specifies the size in bytes of this data structure.

**nVersion (2 bytes):** A 16-bit integer that MUST be set to 0x0001.

**dwFlags (4 bytes):** A set of bit flags that specify properties of the pixel buffer. The properties are not mutually exclusive; any combination of flags is allowed, except where noted otherwise.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P	G	S	X	D	D	S	D	0	0	0	G	S	S	S	S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	O		M	W	B						A	L	Y	E	P																

The following bit flag constants are defined:

Value	Description
DB PFD_DOUBLEBUFFER	The buffer is double-buffered.
S PFD_STEREO	The buffer is <b>stereoscopic</b> . <a href="#">&lt;28&gt;</a>



Value	Description
DW PFD_DRAW_TO_WINDOW	The buffer can draw to a window or device surface.
DM PFD_DRAW_TO_BITMAP	The buffer can draw to a memory bitmap.
X	This bit SHOULD NOT be used.
SO PFD_SUPPORT_OPENGL	The buffer supports OpenGL drawing. See <a href="#">[OPENGL]</a> for more information.
GF PFD_GENERIC_FORMAT	The pixel format is natively supported by the operating system; this is known as the generic implementation. If this bit is clear, the pixel format is supported by a device driver or hardware. <a href="#">&lt;29&gt;</a>
P PFD_NEED_PALETTE	The buffer uses RGBA pixels on a palette-managed device. A metafile <a href="#">LogPalette</a> object, specified in section <a href="#">2.2.15</a> , is required to achieve the best results for this pixel type. Colors in the palette SHOULD be specified according to the values of the <b>cRedBits</b> , <b>cRedShift</b> , <b>cGreenBits</b> , <b>cGreenShift</b> , <b>cBlueBits</b> , and <b>cBlueShift</b> members.
SP PFD_NEED_SYSTEM_PALETTE	Defined in the pixel format descriptors of hardware that supports one hardware palette in 256-color mode only. For such systems to use hardware acceleration, the hardware palette MUST be in a fixed order (for example, 3-3-2) when in RGBA mode, or MUST match the metafile LogPalette object when in color-index mode.
SE PFD_SWAP_EXCHANGE	Specifies the content of the back buffer in the double-buffered main color plane following a buffer swap. Swapping the color buffers causes the exchange of the back buffer's content with the front buffer's content.
SY PFD_SWAP_COPY	Specifies the content of the back buffer in the double-buffered main color plane following a buffer swap. Swapping the color buffers causes the content of the back buffer to be copied to the front buffer. The content of the back buffer is not affected by the swap.
SL PFD_SWAP_LAYER_BUFFERS	Indicates whether a device can swap individual layer planes with pixel formats that include double-buffered overlay or underlay planes. Otherwise all layer planes are swapped together as a group.
GA PFD_GENERIC_ACCELERATED	The pixel format is supported by a device driver that accelerates the generic implementation. If this flag is clear and the <b>PFD_GENERIC_FORMAT</b> flag is set, the pixel format is supported by the generic implementation only.

**iPixelFormat (1 byte):** Specifies the type of pixel data.

Value	Meaning
PFD_TYPE_RGBA 0x00	Each pixel has four components, in this order: red, green, blue, and alpha. The "alpha" component defines the alpha transparency value.

Value	Meaning
PFD_TYPE_COLORINDEX 0x01	Each pixel uses a color-index value.

**cColorBits (1 byte):** Specifies the number of color bitplanes in each color buffer. For RGBA pixel types, it is the size of the color buffer, excluding the alpha bitplanes. For color-index pixels, it is the size of the color-index buffer.

**cRedBits (1 byte):** Specifies the number of red bitplanes in each RGBA color buffer.

**cRedShift (1 byte):** Specifies the shift count in bits for red bitplanes in each RGBA color buffer.

**cGreenBits (1 byte):** Specifies the number of green bitplanes in each RGBA color buffer.

**cGreenShift (1 byte):** Specifies the shift count for green bitplanes in each RGBA color buffer.

**cBlueBits (1 byte):** Specifies the number of blue bitplanes in each RGBA color buffer.

**cBlueShift (1 byte):** Specifies the shift count for blue bitplanes in each RGBA color buffer.

**cAlphaBits (1 byte):** Specifies the number of alpha bitplanes in each RGBA color buffer. [<30>](#)

**cAlphaShift (1 byte):** Specifies the shift count for alpha bitplanes in each RGBA color buffer. [<31>](#)

**cAccumBits (1 byte):** Specifies the total number of bitplanes in the accumulation buffer.

**cAccumRedBits (1 byte):** Specifies the number of red bitplanes in the accumulation buffer.

**cAccumGreenBits (1 byte):** Specifies the number of green bitplanes in the accumulation buffer.

**cAccumBlueBits (1 byte):** Specifies the number of blue bitplanes in the accumulation buffer.

**cAccumAlphaBits (1 byte):** Specifies the number of alpha bitplanes in the accumulation buffer. [<32>](#)

**cDepthBits (1 byte):** Specifies the depth of the depth (z-axis) buffer.

**cStencilBits (1 byte):** Specifies the depth of the stencil buffer.

**cAuxBuffers (1 byte):** Specifies the number of auxiliary buffers. Auxiliary buffers are not supported.

**iLayerType (1 byte):** This field MAY be ignored. [<33>](#)

**bReserved (1 byte):** Specifies the number of overlay and underlay planes. Bits 0 through 3 specify up to 15 overlay planes and bits 4 through 7 specify up to 15 underlay planes.

**dwLayerMask (4 bytes):** This field MAY be ignored. [<34>](#)

**dwVisibleMask (4 bytes):** Specifies the transparent color or index of an underlay plane. When the pixel type is RGBA, **dwVisibleMask** is a transparent RGB color value. When the pixel type is color index, it is a transparent index value.

**dwDamageMask (4 bytes):** This field MAY be ignored. [<35>](#)

### 2.2.21 RegionData Object

The Enhanced Metafile Format **RegionData** object specifies data that defines a region, which is made of non-overlapping rectangles.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RegionDataHeader																															
...																															
...																															
...																															
...																															
...																															
Data (variable)																															
...																															

**RegionDataHeader (32 bytes):** A 256-bit [RegionDataHeader](#) object that describes the following data.

**Data (variable):** An array of [Windows Metafile Format RectL](#) objects, as specified in [\[MS-WMF\]](#) section 2.2.1.14, which are merged together to create the region.

### 2.2.22 RegionDataHeader Object

The Enhanced Metafile Format **RegionDataHeader** object describes the properties of a [RegionData](#) object.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Size																															
Type																															
CountRects																															
RgnSize																															
Bounds																															
...																															
...																															
...																															

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this object in bytes. This MUST be 0x00000020.

**Type (4 bytes):** A 32-bit unsigned integer that specifies the region type. This SHOULD be **RDH\_RECTANGLES** (0x00000001).

**CountRects (4 bytes):** A 32-bit unsigned integer that specifies the number of rectangles in this region.

**RgnSize (4 bytes):** A 32-bit unsigned integer that specifies the size of the buffer of rectangles in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format RectL](#) object, as specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounds of the region.

### 2.2.23 TriVertex Object

The Enhanced Metafile Format **TriVertex** object specifies color and position information for the definition of a rectangle or triangle vertex.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
x																															
y																															
Red																Green															
Blue																Alpha															

**x (4 bytes):** A 32-bit signed integer that specifies the horizontal position in logical units.

**y (4 bytes):** A 32-bit signed integer that specifies the vertical position in logical units.

**Red (2 bytes):** A 16-bit unsigned integer that specifies the red color value for the point.

**Green (2 bytes):** A 16-bit unsigned integer that specifies the green color value for the point.

**Blue (2 bytes):** A 16-bit unsigned integer that specifies the blue color value for the point.

**Alpha (2 bytes):** A 16-bit unsigned integer that specifies the alpha transparency value for the point.

## 2.2.24 UniversalFontId Object

The Enhanced Metafile Format (EMF) UniversalFontId defines a mechanism for identifying font objects in EMF metafiles.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Checksum																															
Index																															

**Checksum (4 bytes):** A 32-bit unsigned integer that is the checksum of the font. The checksum value has the following meanings:

Value	Meaning
0x00000000	The object is a device font.
0x00000001	The object is a <b>Type 1 font</b> that has been installed on the client machine and is enumerated by the PostScript <b>printer driver</b> as a device font.
0x00000002	The object is not a font but is a Type 1 rasterizer.
3 ≤ value	The object is a bitmap, vector, TrueType or Type 1 font rasterized by a Type 1 rasterizer. The checksum value MUST be computed.

If the checksum value is computed, it MUST be computed using the following algorithm:

```
ULONG ComputeFileviewCheckSum(PVOID pvView, ULONG cjView)
{
    ULONG sum;
    PULONG pulCur,pulEnd;

    pulCur = (PULONG) pvView;

    for (sum = 0, pulEnd = pulCur + cjView / sizeof(ULONG);
        pulCur < pulEnd; pulCur += 1)
    {
        sum += 256 * sum + *pulCur;
    }
    return ( sum < 2 ) ? 2 : sum;
}
```

**pvView:** A pointer to the start of the font file.

**cjView:** The length of the font file in bytes.

**Index (4 bytes):** A 32-bit unsigned integer that is an index associated with the font object. The meaning of this field is determined by the type of font.

## 2.2.25 XForm Object

The Enhanced Metafile Format XForm object defines a world-space to page-space transformation. For more information, see [\[MSDN-WRLDPGSPC\]](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
M11																															
M12																															
M21																															
M22																															
Dx																															
Dy																															

**M11 (4 bytes):** A 32-bit float whose definition is dependent upon the operation being applied.

Operation	Meaning
Scaling	Horizontal scaling component

Operation	Meaning
Rotation	Cosine of rotation angle
Reflection	Horizontal component

**M12 (4 bytes):** A 32-bit float whose definition is dependent upon the operation being applied.

Operation	Meaning
Shear	Horizontal proportionality constant
Rotation	Sine of rotation angle

**M21 (4 bytes):** A 32-bit float whose definition is dependent upon the operation being applied.

Operation	Meaning
Shear	Vertical proportionality constant
Rotation	Negative sine of rotation angle

**M22 (4 bytes):** A 32-bit float whose definition is dependent upon the operation being applied.

Operation	Meaning
Scaling	Vertical scaling component
Rotation	Cosine of rotation angle
Reflection	Vertical component

**Dx (4 bytes):** A 32-bit float that contains the horizontal translation component, in logical units.

**Dy (4 bytes):** A 32-bit float that contains the vertical translation component, in logical units.

The following list describes how the members are used for each operation.

Operation	M11	M12	M21	M22
Rotation	Cosine	Sine	Negative sine	Cosine
Scaling	Horizontal scaling component	Not used	Not used	Vertical Scaling Component
Shear	Not used	Horizontal Proportionality Constant	Vertical Proportionality Constant	Not used
Reflection	Horizontal Reflection Component	Not used	Not used	Vertical Reflection Component

## 2.3 EMF Records

This section specifies the Enhanced Metafile Format (EMF) records, which have been grouped into the following general categories:

Name	Section	Description
Bitmap record types	<a href="#">2.3.1</a>	Manage and output bitmap images.
Clipping record types	<a href="#">2.3.2</a>	Specify and manage clipping regions.
Comment record types	<a href="#">2.3.3</a>	Define formats for specifying arbitrary private data, embedding records in other metafile formats, and adding new or special-purpose commands.
Control record types	<a href="#">2.3.4</a>	Define the start and end of an EMF metafile and properties of the metafile.
Drawing record types	<a href="#">2.3.5</a>	Perform graphics drawing.
Escape record types	<a href="#">2.3.6</a>	Execute printer driver functions.
Object creation record types	<a href="#">2.3.7</a>	Create graphics objects.
Object manipulation record types	<a href="#">2.3.8</a>	Manage and modify graphics objects.
OpenGL record types	<a href="#">2.3.9</a>	Specify enhanced metafile records generated by OpenGL.
Path bracket record types	<a href="#">2.3.10</a>	Specify and manipulate paths in path brackets.
State record types	<a href="#">2.3.11</a>	Specify and manage graphics properties.
Transform record types	<a href="#">2.3.12</a>	Specify and modify <b>world space</b> to <b>page space transforms</b> .

### 2.3.1 Bitmap Record Types

The Enhanced Metafile Format (EMF) bitmap record types manage and output bitmap images.

The following are EMF bitmap record types:

Name	Section	Description
EMR_ALPHABLEND	<a href="#">2.3.1.1</a>	Performs a block transfer of pixels from a source bitmap, including alpha transparency data, to the playback device context, according to a specified blending operation.
EMR_BITBLT	<a href="#">2.3.1.2</a>	Performs a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern, to the playback device context, according to a specified raster operation.
EMR_MASKBLT	<a href="#">2.3.1.3</a>	Performs a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern and with the application of a color mask bitmap, to the playback device context, according to specified foreground and background raster



Name	Section	Description
		operations.
EMR_PLGBLT	<a href="#">2.3.1.4</a>	Performs a block transfer of pixels from a source bitmap, with the application of a color mask bitmap, to a parallelogram in the playback device context.
EMR_SETDIBITSTODEVICE	<a href="#">2.3.1.5</a>	Performs a block transfer of pixels from specified scanlines of a source bitmap to the playback device context.
EMR_STRETCHBLT	<a href="#">2.3.1.6</a>	Performs a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern, to the playback device context, according to a specified raster operation, stretching or compressing the output to fit the dimensions of the destination, if necessary.
EMR_STRETCHDIBITS	<a href="#">2.3.1.7</a>	Performs a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern, to the playback device context, according to a specified raster operation, stretching or compressing the output to fit the dimensions of the destination, if necessary.
EMR_TRANSPARENTBLT	<a href="#">2.3.1.8</a>	Performs a block transfer of pixels from a source bitmap, treating a specified color as transparent, to the playback device context, stretching or compressing the output to fit the dimensions of the destination, if necessary.

The generic structure of EMF bitmap records is specified as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
dParm (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that defines the type of record. The bitmap record types are listed below. See the table above for descriptions of these record types.

Name	Value
EMR_BITBLT	0x0000004C
EMR_STRETCHBLT	0x0000004D
EMR_MASKBLT	0x0000004E
EMR_PLGBLT	0x0000004F

Name	Value
EMR_SETDIBITSTODEVICE	0x00000050
EMR_STRETCHDIBITS	0x00000051
EMR_ALPHABLEND	0x00000072
EMR_TRANSPARENTBLT	0x00000074

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size, in bytes, of the **dParm** field in the record.

**dParm (variable):** An array of 32-bit integers that contains parameters for the specific type of bitmap record.

The following notes generally apply to EMF metafile bitmap block transfers, unless noted otherwise:

- The source bitmap header MUST be in [DeviceIndependentBitmap \(DIB\)](#) format. See [\[MS-WMF\]](#) section 2.2.2.3 for more information. The properties that describe the destination bitmap are defined in the playback device context.
- If the color formats of the source and destination do not match, the source bits MUST be converted to the destination format as a part of processing this record.
- If the source and destination rectangles are not the same size, the source bitmap MUST be stretched or compressed to match the destination rectangle. The stretching mode in the playback device context is specified by a value from the [StretchMode](#) enumeration, specified in section [2.1.32](#). It MUST be used to determine how to stretch or compress the pixels, if necessary.

### 2.3.1.1 EMR\_ALPHABLEND Record

The Enhanced Metafile Format (EMF) EMR\_ALPHABLEND record specifies a block transfer of pixels from a source bitmap, including alpha transparency data, to the playback device context, according to a specified blending operation.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															

xDest
yDest
cxDest
cyDest
BLENDFUNCTION
xSrc
ySrc
XformSrc
...
...
...
...
...
BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
cxSrc
cySrc

BmiSrc (variable)
...
BitsSrc (variable)
...

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as EMR\_ALPHABLEND. This MUST be 0x00000072.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format \(WMF\) RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle in device units.

**xDest (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

**yDest (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

**cxDest (4 bytes):** A 32-bit signed integer that specifies the logical width of the destination rectangle. This value MUST be greater than 0.

**cyDest (4 bytes):** A 32-bit signed integer that specifies the logical height of the destination rectangle. This value MUST be greater than 0.

**BLENDFUNCTION (4 bytes):** The **BLENDFUNCTION** structure controls blending by specifying the blending operations for source and destination bitmaps.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
BlendOperation										BlendFlags										SrcConstantAlpha										AlphaFormat									

**BlendOperation (1 byte):** The blend operation code. The only source and destination blend operation that has been defined is 0, which specifies that the source bitmap MUST be combined with the destination bitmap based on the alpha transparency values of the source pixels. See the equations below for details.

**BlendFlags (1 byte):** This value is not used and MUST be 0x00.

**SrcConstantAlpha (1 byte):** An 8-bit unsigned alpha transparency value that determines the blend of the source and destination bitmaps. This value MUST be used on the entire source bitmap. The minimum alpha transparency value, 0, corresponds to completely transparent; and the maximum value, 0xFF, corresponds to completely opaque. In effect, a value of 0xFF specifies that the per-pixel alpha values determine the blend of the source and destination bitmaps. See the equations below for details.

**AlphaFormat (1 byte):** The alpha format, which controls the way the source and destination bitmaps are interpreted.

Value	Meaning
0x00	The pixels in the source bitmap do not specify alpha transparency. In this case, the <b>SrcConstantAlpha</b> value MUST determine the blend of the source and destination bitmaps. Note that in the equations below <b>SrcConstantAlpha</b> is divided by 255, which produces a value in the range 0 to 1.
AC_SRC_ALPHA 0x01	Indicates that the source bitmap MUST be 32 bits per pixel and MUST specify an alpha transparency value for each pixel.

**xSrc (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the source rectangle.

**ySrc (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the source rectangle.

**XformSrc (24 bytes):** A 192-bit [XForm](#) object, specified in section [2.2.25](#), which specifies a two-dimensional, linear world-space to page-space transformation for the source bitmap. For more information on coordinate spaces and transformations, see [\[MSDN-WRLDPGSPC\]](#).

**BkColorSrc (4 bytes):** A 32-bit WMF **ColorRef** object, specified in [\[MS-WMF\]](#) section 2.2.1.7, which specifies the background RGB color of the source bitmap.

**UsageSrc (4 bytes):** A 32-bit unsigned integer that specifies how to interpret color values in the source bitmap. The value MUST be in the [DIBColors](#) enumeration, specified in section [2.1.9](#).

**offBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap header.

**cbBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap header.

**offBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap bits.

**cbBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap bits.

**cxSrc (4 bytes):** A 32-bit signed integer that specifies the logical width of the source rectangle. This value MUST be greater than 0.

**cySrc (4 bytes):** A 32-bit signed integer that specifies the logical height of the source rectangle. This value MUST be greater than 0.

**BmiSrc (variable):** A buffer containing the source bitmap header.

**BitsSrc (variable):** A buffer containing the source bitmap bits.

This record MAY perform scaling, translation or reflection transformations, according to the **XformSrc** field; but rotation and shear transformations MUST not occur.

The equations below show how destination pixels are computed from source pixels using **BLENDFUNCTION**. In the equations, "dst" refers to the destination bitmap, and "src" refers to the source bitmap. The color and transparency values of the source and destination pixels are denoted by "Red", "Green", "Blue", and "Alpha".

**Case I:** The **AlphaFormat** value is 0, which means the **SrcConstantAlpha** value MUST be used to blend the source and destination bitmaps, as follows:

```
dst.Red = src.Red * (SrcConstantAlpha/255.0) +  
          dst.Red * (1.0 - (SrcConstantAlpha/255.0))  
  
dst.Green = src.Green * (SrcConstantAlpha/255.0) +  
            dst.Green * (1.0 - (SrcConstantAlpha/255.0))  
  
dst.Blue = src.Blue * (SrcConstantAlpha/255.0) +  
            dst.Blue * (1.0 - (SrcConstantAlpha/255.0))
```

If the destination bitmap has an alpha channel, then it is blended as follows:

```
dst.Alpha = src.Alpha * (SrcConstantAlpha/255.0) +  
            dst.Alpha * (1.0 - (SrcConstantAlpha/255.0))
```

Note that if **SrcConstantAlpha** is 0xFF, these equations reduce to a simple source copy to the destination.

**Case II:** The **AlphaFormat** value is **AC\_SRC\_ALPHA**, which means the source pixels MUST be pre-multiplied by **SrcConstantAlpha**, and then the blend MUST be based on the per-pixel source alpha channel, as follows:

```
src.Red = src.Red * (SrcConstantAlpha/255.0)  
src.Green = src.Green * (SrcConstantAlpha/255.0)  
src.Blue = src.Blue * (SrcConstantAlpha/255.0)  
  
dst.Red = src.Red + (1.0 - (src.Alpha/255.0)) * dst.Red  
dst.Green = src.Green + (1.0 - (src.Alpha/255.0)) * dst.Green  
dst.Blue = src.Blue + (1.0 - (src.Alpha/255.0)) * dst.Blue
```

If the destination bitmap has an alpha channel, then it is blended as follows:

```
src.Alpha = src.Alpha * (SrcConstantAlpha)/255.0  
  
dst.Alpha = src.Alpha + (1.0 - (src.Alpha/255.0)) * dst.Alpha
```

Note that if **SrcConstantAlpha** is 0xFF, there is in effect no premultiplication of the source values.

See section [2.3.1](#) for the specification of other Bitmap record types.

### 2.3.1.2 EMR\_BITBLT Record

The Enhanced Metafile Format (EMF) EMR\_BITBLT record specifies a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern, to the playback device context, according to a specified raster operation.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
xDest																															
yDest																															
cxDest																															
cyDest																															
BitBltRasterOperation																															
xSrc																															
ySrc																															
XformSrc																															
...																															
...																															
...																															

...
...
BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
BmiSrc (variable)
...
BitsSrc (variable)
...

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as EMR\_BITBLT. This MUST be 0x0000004C.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format \(WMF\) RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle in device units.

**xDest (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

**yDest (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

**cxDest (4 bytes):** A 32-bit signed integer that specifies the logical width of the source and destination rectangles.

**cyDest (4 bytes):** A 32-bit signed integer that specifies the logical height of the source and destination rectangles.

**BitBltRasterOperation (4 bytes):** A 32-bit unsigned integer that specifies the raster operation code. This code defines how the color data of the source rectangle is to be combined with the color data of the destination rectangle and optionally a brush pattern, to achieve the final color.



The value MUST be in the WMF [Ternary Raster Operation](#) enumeration, as specified in [\[MS-WMF\]](#) section **2.1.33**.

**xSrc (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the source rectangle.

**ySrc (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the source rectangle.

**XformSrc (24 bytes):** A 192-bit [XForm](#) object, specified in section [2.2.25](#), which specifies a two-dimensional, linear world-space to page-space transformation of the source bitmap. For more information on coordinate spaces and transformations, see [\[MSDN-WRLDPGSPC\]](#).

**BkColorSrc (4 bytes):** A 32-bit WMF [ColorRef](#) object, as specified in [\[MS-WMF\]](#) section 2.2.1.7, which specifies the background RGB color of the source bitmap.

**UsageSrc (4 bytes):** A 32-bit unsigned integer that specifies how to interpret color values in the source bitmap. The value MUST be in the [DIBColors](#) enumeration, specified in section [2.1.9](#).

**offBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap header.

**cbBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap header.

**offBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap bits.

**cbBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap bits.

**BmiSrc (variable):** An optional buffer containing the source bitmap header. A bitmap header is not required if a source is not used in the specified raster operation.

**BitsSrc (variable):** An optional buffer containing the source bitmap bits. The bitmap bits are not required if a source is not used in the specified raster operation.

This record MAY perform scaling, translation or reflection transformations, according to the **XformSrc** field; but rotation and shear transformations MUST not occur.

See section [2.3.1](#) for the specification of other Bitmap record types.

### 2.3.1.3 EMR\_MASKBLT Record

The Enhanced Metafile Format (EMF) EMR\_MASKBLT record specifies a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern and with the application of a color mask bitmap, to the playback device context, according to specified foreground and background raster operations.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															

Size
Bounds
...
...
...
xDest
yDest
cxDest
cyDest
ROP4
xSrc
ySrc
XformSrc
...
...
...
...
...
BkColorSrc
UsageSrc
offBmiSrc

cbBmiSrc
offBitsSrc
cbBitsSrc
xMask
yMask
UsageMask
offBmiMask
cbBmiMask
offBitsMask
cbBitsMask
BmiSrc (variable)
...
BitsSrc (variable)
...
BmiMask (variable)
...
BitsMask (variable)
...

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_MASKBLT**. This MUST be 0x0000004E.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format \(WMF\) RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle in device units.

- xDest (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.
- yDest (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.
- cxDest (4 bytes):** A 32-bit signed integer that specifies the logical width of the destination rectangle.
- cyDest (4 bytes):** A 32-bit signed integer that specifies the logical height of the destination rectangle.
- ROP4 (4 bytes):** A quaternary raster operation, which specifies ternary raster operations for the foreground and background colors of a bitmap.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reserved																BackgroundROP3								ForegroundROP3							

- Reserved (2 bytes):** This field MUST be 0x0000 and MUST be ignored.
- BackgroundROP3 (1 byte):** The unsigned, most-significant 8-bits of a 24-bit ternary raster operation value from the WMF [Ternary Raster Operation](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.33**. This code defines how to combine the background color data of the source and destination bitmaps and brush pattern.
- ForegroundROP3 (1 byte):** The unsigned, most-significant 8 bits of a 24-bit ternary raster operation value from the WMF **Ternary Raster Operation** enumeration. This code defines how to combine the foreground color data of the source and destination bitmaps and brush pattern.
- xSrc (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the source rectangle.
- ySrc (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the source rectangle.
- XformSrc (24 bytes):** A 192-bit [XForm](#) object, specified in section [2.2.25](#), that specifies a world-space to page-space transformation. For more information on coordinate spaces and transformations, see [\[MSDN-WRLDPGSPC\]](#).
- BkColorSrc (4 bytes):** A 32-bit WMF [ColorRef](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.7, which specifies the background color of the source rectangle.
- UsageSrc (4 bytes):** A 32-bit unsigned integer that specifies how to interpret color values in the source bitmap. The value MUST be in the [DIBColors](#) enumeration, specified in section [2.1.9](#).
- offBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap header.
- cbBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap header.

**offBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap bits.

**cbBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap bits.

**xMask (4 bytes):** A 32-bit signed integer that specifies the horizontal pixel offset into the mask bitmap.

**yMask (4 bytes):** A 32-bit signed integer that specifies the vertical pixel offset into the mask bitmap.

**UsageMask (4 bytes):** A 32-bit unsigned integer that specifies how to interpret color values in the mask bitmap. The value **MUST** be in the **DIBColors** enumeration.

**offBmiMask (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the mask bitmap header.

**cbBmiMask (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the mask bitmap header.

**offBitsMask (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the mask bitmap bits.

**cbBitsMask (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the mask bitmap bits.

**BmiSrc (variable):** A buffer containing the source bitmap header.

**BitsSrc (variable):** A buffer containing the source bitmap bits.

**BmiMask (variable):** A buffer containing the mask bitmap header.

**BitsMask (variable):** A buffer containing the mask bitmap bits.

This record **MAY** perform scaling, translation or reflection transformations, according to the **XformSrc** field; but rotation and shear transformations **MUST** not be performed.

The mask bitmap **MUST** be a monochrome bitmap. A value of 1 in the mask indicates that the color of the corresponding source pixel **SHOULD** be copied to the destination. A value of 0 in the mask indicates that the destination pixel color **SHOULD NOT** be changed. If the mask rectangle is smaller than the source and destination rectangles, the mask pattern **MUST** be replicated as necessary.

The source and mask bitmap fields **MAY** be in the record in any order.

See section [2.3.1](#) for the specification of other Bitmap record types.

#### 2.3.1.4 EMR\_PLGBLT Record

The Enhanced Metafile Format (EMF) EMR\_PLGBLT record specifies a block transfer of pixels from a source bitmap, with the application of a color mask bitmap, to a parallelogram in the playback device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
aptIDest																															
...																															
...																															
...																															
xSrc																															
ySrc																															
cxSrc																															
cySrc																															
XformSrc																															
...																															
...																															

...
...
...
BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
xMask
yMask
UsageMask
offBmiMask
cbBmiMask
offBitsMask
cbBitsMask
BmiSrc (variable)
...
BitsSrc (variable)
...
BmiMask (variable)

...
BitsMask (variable)
...

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_PLGBLT**. This MUST be 0x0000004F.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format \(WMF\) RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle in device units.

**aptlDest (24 bytes):** A 192-bit array of three WMF [PointL](#) objects, specified in [\[MS-WMF\]](#) section 2.2.1.11, which specify three corners of the destination parallelogram.

The upper-left corner of the source rectangle is mapped to the first point in this array, the upper-right corner to the second point, and the lower-left corner to the third point. The lower-right corner of the source rectangle is mapped to the implicit fourth point in the parallelogram (see below).

**xSrc (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the source rectangle.

**ySrc (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the source rectangle.

**cxSrc (4 bytes):** A 32-bit signed integer that specifies the logical width of the source rectangle.

**cySrc (4 bytes):** A 32-bit signed integer that specifies the logical height of the source rectangle.

**XformSrc (24 bytes):** A 192-bit [XForm](#) object, specified in section [2.2.25](#), that specifies a two-dimensional, linear world-space to page-space scaling, translation or reflection transformation of the source bitmap. For more information on coordinate spaces and transformations, see [\[MSDN-WRLDPGSPC\]](#).

**BkColorSrc (4 bytes):** A 32-bit WMF [ColorRef](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.7, which specifies the background RGB color of the source bitmap.

**UsageSrc (4 bytes):** A 32-bit unsigned integer that specifies how to interpret color values in the source bitmap. The value MUST be in the [DIBColors](#) enumeration, specified in section [2.1.9](#).

**offBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap header.

**cbBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap header.

**offBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap bits.



**cbBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap bits.

**xMask (4 bytes):** A 32-bit signed integer that specifies the horizontal pixel offset into mask bitmap.

**yMask (4 bytes):** A 32-bit signed integer that specifies the vertical pixel offset into mask bitmap.

**UsageMask (4 bytes):** A 32-bit unsigned integer that specifies how to interpret color values in the mask bitmap. The value MUST be in the **DIBColors** enumeration.

**offBmiMask (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the mask bitmap header.

**cbBmiMask (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the mask bitmap header.

**offBitsMask (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the mask bitmap bits.

**cbBitsMask (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the mask bitmap bits.

**BmiSrc (variable):** A buffer containing the source bitmap header.

**BitsSrc (variable):** A buffer containing the source bitmap bits.

**BmiMask (variable):** A buffer containing the mask bitmap header.

**BitsMask (variable):** A buffer containing the mask bitmap bits.

The fourth vertex of the parallelogram (D) is defined by treating the first three points (A, B, and C) as vectors and computing  $D = B + C - A$ .

This record MAY perform scaling, translation or reflection transformations, according to the **XformSrc** field; but rotation and shear transformations MUST not be performed.

The mask bitmap MUST be a monochrome bitmap. A value of 1 in the mask indicates that the color of the corresponding source pixel SHOULD be copied to the destination. A value of 0 in the mask indicates that the destination pixel color SHOULD NOT be changed. If the mask rectangle is smaller than the source and destination rectangles, the mask pattern MUST be replicated as necessary.

The source and mask bitmap fields MAY be in the record in any order.

See section [2.3.1](#) for the specification of other Bitmap record types.

### 2.3.1.5 EMR\_SETDIBITSTODEVICE Record

The Enhanced Metafile Format (EMF) EMR\_SETDIBITSTODEVICE record specifies a block transfer of pixels from specified scanlines of a source bitmap to the playback device context.

This record supports source images in **JPEG** or **PNG** formats.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
xDest																															
yDest																															
xSrc																															
ySrc																															
cxSrc																															
cySrc																															
offBmiSrc																															
cbBmiSrc																															
offBitsSrc																															
cbBitsSrc																															
UsageSrc																															
iStartScan																															
cScans																															

BmiSrc (variable)
...
BitsSrc (variable)
...

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETDIBITSTODEVICE**. This MUST be 0x00000050.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format \(WMF\) RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle in device units.

**xDest (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

**yDest (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

**xSrc (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the lower-left corner of the source rectangle.

**ySrc (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the lower-left corner of the source rectangle.

**cxSrc (4 bytes):** A 32-bit signed integer that specifies the logical width of the source rectangle.

**cySrc (4 bytes):** A 32-bit signed integer that specifies the logical height of the source rectangle.

**offBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap header.

**cbBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap header.

**offBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap bits.

**cbBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap bits.

**UsageSrc (4 bytes):** A 32-bit unsigned integer that specifies how to interpret color values in the source bitmap. The value MUST be in the [DIBColors](#) enumeration, specified in section [2.1.9](#).

**iStartScan (4 bytes):** A 32-bit unsigned integer that specifies the first scan line in the array.

**cScans (4 bytes):** A 32-bit unsigned integer that specifies the number of scan lines.

**BmiSrc (variable):** A buffer containing the source bitmap header.

**BitsSrc (variable):** A buffer containing the source bitmap bits.

If the **Compression** field in the source DIB header is set to **BI\_JPEG** or **BI\_PNG**, the source bitmap bits MUST contain a JPEG or PNG image, respectively.

See section [2.3.1](#) for the specification of other Bitmap record types.

**2.3.1.6 EMR\_STRETCHBLT Record**

The Enhanced Metafile Format (EMF) EMR\_STRETCHBLT record specifies a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern, to the playback device context, according to a specified raster operation, stretching or compressing the output to fit the dimensions of the destination, if necessary.

The expansion or compression of the bitmap MUST be performed according to the [StretchMode](#) enumeration value that is currently set in the playback device context.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
xDest																															
yDest																															
cxDest																															
cyDest																															
BitBltRasterOperation																															
xSrc																															
ySrc																															

XformSrc
...
...
...
...
...
BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
cxSrc
cySrc
BmiSrc (variable)
...
BitsSrc (variable)
...

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_STRETCHBLT**. This MUST be 0x0000004D.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format \(WMF\) RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle in device units.

**xDest (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

**yDest (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

**cxDest (4 bytes):** A 32-bit signed integer that specifies the logical width of the destination rectangle.

**cyDest (4 bytes):** A 32-bit signed integer that specifies the logical height of the destination rectangle.

**BitBltRasterOperation (4 bytes):** A 32-bit unsigned integer that specifies the raster operation code. This code defines how the color data of the source rectangle is to be combined with the color data of the destination rectangle and optionally a brush pattern, to achieve the final color.

The value MUST be in the WMF **Ternary Raster Operation** enumeration, specified in [\[MS-WMF\]](#) section 2.1.33.

**xSrc (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the source rectangle.

**ySrc (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the source rectangle.

**XformSrc (24 bytes):** A 192-bit [XForm](#) object, specified in section 2.2.25, which specifies a two-dimensional, linear world-space to page-space transformation of the source bitmap. For more information on coordinate spaces, see [\[MSDN-WRLDPGSPC\]](#).

**BkColorSrc (4 bytes):** A 32-bit WMF [ColorRef](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.7, which specifies the background RGB color of the source bitmap.

**UsageSrc (4 bytes):** A 32-bit unsigned integer that specifies how to interpret color values in the source bitmap. The value MUST be in the [DIBColors](#) enumeration, specified in section 2.1.9.

**offBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap header.

**cbBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap header.

**offBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap bits.

**cbBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap bits.

**cxSrc (4 bytes):** A 32-bit signed integer that specifies the logical width of the source rectangle.

**cySrc (4 bytes):** A 32-bit signed integer that specifies the logical height of the source rectangle.

**BmiSrc (variable):** An optional buffer containing the source bitmap header. A bitmap header is not required if a source is not used in the specified raster operation.

**BitsSrc (variable):** An optional buffer containing the source bitmap bits. The bitmap bits are not required if a source is not used in the specified raster operation.

This record MAY perform scaling, translation or reflection transformations, according to the **XformSrc** field; but rotation and shear transformations MUST not occur.

See section [2.3.1](#) for the specification of other Bitmap record types.

**2.3.1.7 EMR\_STRETCHDIBITS Record**

The Enhanced Metafile Format (EMF) EMR\_STRETCHDIBITS record specifies a block transfer of pixels from a source bitmap, optionally in combination with a brush pattern, to the playback device context, according to a specified raster operation, stretching or compressing the output to fit the dimensions of the destination, if necessary.

The expansion or compression of the bitmap MUST be performed according to the [StretchMode](#) enumeration value that is currently set in the playback device context.

This record supports source images in JPEG or PNG formats.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
xDest																															
yDest																															
xSrc																															
ySrc																															
cxSrc																															
cySrc																															

offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
UsageSrc
BitBltRasterOperation
cxDest
cyDest
BmiSrc (variable)
...
BitsSrc (variable)
...

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_STRETCHDIBITS**. This MUST be 0x00000051.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format \(WMF\) RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle in device units.

**xDest (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

**yDest (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

**xSrc (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the source rectangle.

**ySrc (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the source rectangle.

**cxSrc (4 bytes):** A 32-bit signed integer that specifies the logical width of the source rectangle.

**cySrc (4 bytes):** A 32-bit signed integer that specifies the logical height of the source rectangle.



**offBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap header.

**cbBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap header.

**offBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap bits.

**cbBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap bits.

**UsageSrc (4 bytes):** A 32-bit unsigned integer that specifies how to interpret color values in the source bitmap. The value MUST be in the [DIBColors](#) enumeration, specified in section [2.1.9](#).

**BitBltRasterOperation (4 bytes):** A 32-bit unsigned integer that specifies a raster operation code. These codes define how the color data of the source rectangle is to be combined with the color data of the destination rectangle and optionally a brush pattern, to achieve the final color.

The value MUST be in the WMF **Ternary Raster Operation** enumeration, specified in [\[MS-WMF\]](#) section **2.1.33**.

**cxDest (4 bytes):** A 32-bit signed integer that specifies the logical width of the destination rectangle.

**cyDest (4 bytes):** A 32-bit signed integer that specifies the logical height of the destination rectangle.

**BmiSrc (variable):** A buffer containing the source bitmap header.

**BitsSrc (variable):** A buffer containing the source bitmap bits.

If the **Compression** field in the source DIB header is set to **BI\_JPEG** or **BI\_PNG**, the source bitmap bits MUST contain a JPEG or PNG image, respectively.

This record specifies a mirror-image copy of the source bitmap to the destination if the signs of the height or width fields differ. That is, if **cxSrc** and **cxDest** have different signs, this record specifies a mirror image of the source bitmap along the x-axis. If **cySrc** and **cyDest** have different signs, this record specifies a mirror image of the source bitmap along the y-axis.

See section [2.3.1](#) for the specification of other Bitmap record types.

### 2.3.1.8 EMR\_TRANSPARENTBLT Record

The Enhanced Metafile Format (EMF) EMR\_TRANSPARENTBLT record specifies a block transfer of pixels from a source bitmap, treating a specified color as transparent, to the playback device context, stretching or compressing the output to fit the dimensions of the destination, if necessary.

The expansion or compression of the bitmap MUST be performed according to the [StretchMode](#) enumeration value that is currently set in the playback device context.



BkColorSrc
UsageSrc
offBmiSrc
cbBmiSrc
offBitsSrc
cbBitsSrc
cxSrc
cySrc
BmiSrc (variable)
...
BitsSrc (variable)
...

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_TRANSPARENTBLT**. This MUST be 0x00000074.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format \(WMF\) RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle in device units.

**xDest (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

**yDest (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

**cxDest (4 bytes):** A 32-bit signed integer that specifies the logical width of the destination rectangle.

**cyDest (4 bytes):** A 32-bit signed integer that specifies the logical height of the destination rectangle.

**TransparentColor (4 bytes):** A 32-bit WMF [ColorRef](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.7, which specifies the color to be treated as transparent in the source data.

**xSrc (4 bytes):** A 32-bit signed integer that specifies the logical x-coordinate of the upper-left corner of the source rectangle.

**ySrc (4 bytes):** A 32-bit signed integer that specifies the logical y-coordinate of the upper-left corner of the source rectangle.

**XformSrc (24 bytes):** A 192-bit [XForm](#) object, specified in section [2.2.25](#), which specifies a two-dimensional, linear world-space to page-space transformation of the source bitmap. For more information on coordinate spaces, see [\[MSDN-WRLDPGSPC\]](#).

**BkColorSrc (4 bytes):** A 32-bit WMF ColorRef object that specifies the background color RGB value of the source bitmap.

**UsageSrc (4 bytes):** A 32-bit unsigned integer that specifies how to interpret color values in the source bitmap. The value MUST be in the [DIBColors](#) enumeration, specified in section [2.1.9](#).

**offBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap header.

**cbBmiSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap header.

**offBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the offset in bytes from the start of this record to the source bitmap bits.

**cbBitsSrc (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the source bitmap bits.

**cxSrc (4 bytes):** A 32-bit signed integer that specifies the logical width of the source rectangle.

**cySrc (4 bytes):** A 32-bit signed integer that specifies the logical height of the source rectangle.

**BmiSrc (variable):** A buffer containing the source bitmap header. [<36>](#)

**BitsSrc (variable):** A buffer containing the source bitmap bits.

See section [2.3.1](#) for the specification of other Bitmap record types.

## 2.3.2 Clipping Record Types

The Enhanced Metafile Format (EMF) clipping record types specify and manage clipping regions.

The following are EMF clipping record types:

Name	Section	Description
EMR_EXCLUDECLIPRECT	<a href="#">2.3.2.1</a>	Specifies a new clipping region that consists of the existing clipping region minus the specified rectangle.
EMR_EXTSELECTCLIPRGN	<a href="#">2.3.2.2</a>	Combines the specified region with the current clip region using the specified mode.
EMR_INTERSECTCLIPRECT	<a href="#">2.3.2.3</a>	Specifies a new clipping region from the intersection of the current clipping region and the specified rectangle.
EMR_OFFSETCLIPRGN	<a href="#">2.3.2.4</a>	Specifies the clipping region with the specified offsets.

Name	Section	Description
EMR_SELECTCLIPPATH	<a href="#">2.3.2.5</a>	Specifies the current path as a clipping region for the playback device context, combining the new region with any existing clipping region using the specified mode.
EMR_SETMETARGN	2.3.2	Intersets the current metaregion with the current clipping region to form a new metaregion for the playback device context. The current clipping region SHOULD be reset to null. This EMF record specifies no parameters.

The generic structure of EMF clipping records is specified as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
dParm (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that defines the type of record. The clipping record types are listed below. See the table above for descriptions of these record types.

Name	Value
EMR_OFFSETCLIPRGN	0x0000001A
EMR_SETMETARGN	0x0000001C
EMR_EXCLUDECLIPRECT	0x0000001D
EMR_INTERSECTCLIPRECT	0x0000001E
EMR_SELECTCLIPPATH	0x00000043
EMR_EXTSELECTCLIPRGN	0x0000004B

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size, in bytes, of the **dParm** field in the record.

**dParm (variable):** An optional array of 32-bit unsigned integers that specify parameters for the clipping record.

### 2.3.2.1 EMR\_EXCLUDECLIPRECT Record

The Enhanced Metafile Format EMR\_EXCLUDECLIPRECT record specifies a new clipping region that consists of the existing clipping region minus the specified rectangle.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Clip																															
...																															
...																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_EXCLUDECLIPRECT**. This MUST be 0x0000001D.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Clip (16 bytes):** A 128-bit [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the clipping rectangle in logical units.

The lower and right edges of the specified rectangle are not excluded from the clipping region.

See section [2.3.2](#) for the specification of other clipping record types.

### 2.3.2.2 EMR\_EXTSELECTCLIPRGN Record

The Enhanced Metafile Format EMR\_EXTSELECTCLIPRGN record combines the specified region with the current clip region using the specified mode.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
RgnDataSize																															
RegionMode																															
RgnData (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_EXTSELECTCLIPRGN**. This MUST be 0x0000004B.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**RgnDataSize (4 bytes):** A 32-bit unsigned integer that specifies the size of region data in bytes.

**RegionMode (4 bytes):** A 32-bit unsigned integer that specifies the way to use the region. The value MUST be in the [RegionMode \(section 2.1.29\)](#) enumeration.

**RgnData (variable):** A **RgnDataSize** length array of bytes that specifies an **RegionData** object in logical units. If **RegionMode** is **RGN\_COPY**, this data can be omitted and the clip region will be set to the default (NULL) clip region.

See section [2.3.2](#) for the specification of other clipping record types.

### 2.3.2.3 EMR\_INTERSECTCLIPRECT Record

The Enhanced Metafile Format EMR\_INTERSECTCLIPRECT record specifies a new clipping region from the intersection of the current clipping region and the specified rectangle.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Clip																															
...																															
...																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_INTERSECTCLIPRECT**. This MUST be 0x0000001E.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Clip (16 bytes):** A 128-bit [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the rectangle in logical units.

The lower and right edges of the specified rectangle are excluded from the clipping region.

See section [2.3.2](#) for the specification of other clipping record types.

#### 2.3.2.4 EMR\_OFFSETCLIPRGN Record

The Enhanced Metafile Format EMR\_OFFSETCLIPRGN record respecifies the clipping region of a playback device context by the specified offsets.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Offset																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_OFFSETCLIPRGN**. This MUST be 0x0000001A.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.



**Offset (8 bytes):** A 64-bit Windows Metafile Format **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies the offset in logical units.

See section [2.3.2](#) for the specification of other clipping record types.

### 2.3.2.5 EMR\_SELECTCLIPPATH Record

The Enhanced Metafile Format EMR\_SELECTCLIPPATH record specifies the current path as a clipping region for a playback device context, combining the new region with any existing clipping region using the specified mode.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
RegionMode																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SELECTCLIPPATH**. This MUST be 0x00000043.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**RegionMode (4 bytes):** A 32-bit unsigned integer that specifies the way to use the path. The value MUST be in the [RegionMode \(section 2.1.29\)](#) enumeration.

See section [2.3.2](#) for the specification of other clipping record types.

### 2.3.3 Comment Record Types

The Enhanced Metafile Format (EMF) comment record types define formats for specifying arbitrary private data, embedding records in other metafile formats, and adding new or special-purpose commands.

The following are EMF comment record types:

Name	Section	Description
EMR_COMMENT	<a href="#">2.3.3.1</a>	Defines a format for specifying arbitrary private data.
EMF_COMMENT_EMFPLUS	<a href="#">2.3.3.2</a>	Contains embedded <b>Enhanced Metafile Format Plus Extensions (EMF+)</b> records.
EMR_COMMENT_EMFSPOOL	<a href="#">2.3.3.3</a>	Contains embedded <b>Enhanced Metafile Spool Format (EMFSPOOL)</b> records.
EMR_COMMENT_PUBLIC	<a href="#">2.3.3.4</a>	Defines a format for adding new or special-purpose commands.

The generic structure of EMF comment records is specified as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
DataSize																															
Identifier (optional)																															
EMFCommentData (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer from the [RecordType enumeration \(section 2.1.1\)](#) that identifies this record as a comment record. This value MUST be 0x00000046.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**DataSize (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the fields that follow.

**Identifier (4 bytes):** An optional, 32-bit unsigned integer that identifies the type of comment record. See the table above for descriptions of these record types.

Valid identifier values are listed below. If this field contains any other value, the comment record MUST be an EMR\_COMMENT record (section 2.3.3.1).

Name	Value
EMR_COMMENT_EMFSPPOOL	0x00000000
EMR_COMMENT_EMFPLUS	0x2B464D45
EMR_COMMENT_PUBLIC	0x43494447

**EMFCommentData (variable):** An array of bytes that specifies the comment data. The length of this array is equal to the value of the **DataSize** field less the size of the **Identifier** field.

### 2.3.3.1 EMR\_COMMENT Record

The Enhanced Metafile Format (EMF) EMR\_COMMENT record is a comment record that defines a format for specifying arbitrary private data.

**Note** Fields that are not described below are specified in [Comment Record Types \(section 2.3.3\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
DataSize																															
Data (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type from the [RecordType enumeration \(section 2.1.1\)](#). It MUST be **EMR\_COMMENT**, which is 0x00000046.

**DataSize (4 bytes):** A 32-bit unsigned integer that specifies the size of private data in the **Data** field in bytes.

**Data (variable):** A **DataSize** length array of bytes that specifies the private data. The first 32 bits of this data MUST NOT be one of the pre-defined comment record type values specified in section [2.3.3](#).

Private data is unknown to EMF; it is meaningful only to applications that know the format of the data and how to use it. EMR\_COMMENT private data records MAY be ignored. [<37>](#)

See section [2.3.3](#) for the specification of other comment record types.

### 2.3.3.2 EMR\_COMMENT\_EMFPLUS Record

The Enhanced Metafile Format (EMF) EMR\_COMMENT\_EMFPLUS record is a comment record that contains embedded Enhanced Metafile Format Plus Extensions (EMF+) records.

**Note** Fields that are not described below are specified in [Comment Record Types \(section 2.3.3\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
DataSize																															
Identifier																															
EMFPlusRecordData (variable)																															
...																															

**Identifier (4 bytes):** A 32-bit unsigned integer that identifies this comment record as containing EMF+ metafile records. The value 0x2B464D45, which is the ASCII string "+FME", identifies this as an EMR\_COMMENT\_EMFPLUS record.

**EMFPlusRecordData (variable):** A variable-length array of bytes that contains one or more EMF+ metafile records, specified in [\[MS-EMFPLUS\]](#) section 2.2. The length of this array is **DataSize**, less the size of the **Identifier** field.

All comment records are types of [EMR\\_COMMENT records \(section 2.3.3.1\)](#). See section [2.3.3](#) for the specification of other comment records.

### 2.3.3.3 EMR\_COMMENT\_EMFSPOOL Record

The Enhanced Metafile Format (EMF) EMR\_COMMENT\_EMFSPOOL record is a comment record that contains embedded Enhanced Metafile Spool Format (EMFSPOOL) records.

**Note** Fields that are not described below are specified in [Comment Record Types \(section 2.3.3\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
DataSize																															
Identifier																															
RecordSignature																															
EMFSpoolFontDefinitionData (variable)																															
...																															

**Identifier (4 bytes):** A 32-bit unsigned integer that identifies this comment record as containing **EMFSPPOOL** metafile records. The value 0x00000000 identifies this as an EMR\_COMMENT\_EMFSPPOOL record.

**RecordSignature (4 bytes):** A 32-bit unsigned integer that identifies the type of EMR\_COMMENT\_EMFSPPOOL record. The value 0x544F4E46, which is the ASCII string "TONF", identifies this as a record that contains embedded EMFSPPOOL font definition data.

**EMFSpoolFontDefinitionData (variable):** A variable-length array of bytes that contains one or more EMFSPPOOL metafile font definition records ([\[MS-EMFSPPOOL\]](#) section 2.2.2.3). The length of this array is **DataSize**, less the sizes of the **Identifier** and **RecordSignature** fields.

All comment records are types of [EMR\\_COMMENT records \(section 2.3.3.1\)](#). See section [2.3.3](#) for the specification of other comment records.

#### 2.3.3.4 EMR\_COMMENT\_PUBLIC Record Types

The Enhanced Metafile Format (EMF) EMR\_COMMENT\_PUBLIC record is a comment record that defines a format for adding special-purpose public data. Public data can specify extensions to EMF metafile processing.

The following are the EMF public comment record types that have been defined:

Name	Section	Description
EMR_COMMENT_BEGINGROUP	<a href="#">2.3.3.4.1</a>	Identifies the beginning of a group of drawing records.
EMR_COMMENT_ENDGROUP	<a href="#">2.3.3.4.2</a>	Identifies the end of a group of drawing records.
EMR_COMMENT_MULTIFORMATS	<a href="#">2.3.3.4.3</a>	Allows multiple definitions of an image to be included in the metafile.

Name	Section	Description
EMR_COMMENT_WINDOW_METAFILE	<a href="#">2.3.3.4.4</a>	Specifies an image in an embedded Windows Metafile Format (WMF) or EMF metafile.

The generic structure of EMF EMR\_COMMENT\_PUBLIC records is specified as follows:

**Note** Fields that are not described below are specified in [Comment Record Types \(section 2.3.3\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
DataSize																															
Identifier																															
CommentType																															
EMFPublicCommentData (variable)																															
...																															

**Identifier (4 bytes):** A 32-bit unsigned integer that identifies this comment record as specifying public data. The value 0x43494447, which is the ASCII string "CIDG", identifies this as an EMR\_COMMENT\_PUBLIC record.

**CommentType (4 bytes):** A 32-bit unsigned integer that identifies the type of EMR\_COMMENT\_PUBLIC record. This SHOULD be one of the values listed in the table above, which are specified in the [EmrComment enumeration \(section 2.1.10\)](#), unless additional EMR\_COMMENT\_PUBLIC record types have been implemented on the **print server**.

**EMFPublicCommentData (variable):** A variable-length array of bytes that contains one or more EMF metafile records, specified in the sections that follow. The length of this array is **DataSize**, less the sizes of the **Identifier** and **CommentType** fields.

All comment records are types of [EMR\\_COMMENT records \(section 2.3.3.1\)](#). See section [2.3.3](#) for the specification of other comment records. Fields that are not described above are specified in that section.

#### 2.3.3.4.1 EMR\_COMMENT\_BEGINGROUP Record

The Enhanced Metafile Format (EMF) EMR\_COMMENT\_BEGINGROUP comment record identifies the beginning of a group of drawing records.

**Note** Fields that are not described below are specified in [EMR\\_COMMENT\\_PUBLIC Record Types \(section 2.3.3.4\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
DataSize																															
Identifier																															
CommentType																															
Rectangle																															
...																															
...																															
...																															
nDescription																															
Description (variable)																															
...																															

**CommentType (4 bytes):** A 32-bit unsigned integer that identifies the type of public comment record, from the [EmrComment enumeration \(section 2.1.10\)](#). It MUST be **EMR\_COMMENT\_BEGINGROUP**, which is 0x00000002.

**Rectangle (16 bytes):** A Windows Metafile Format (WMF) [RectL object](#) ([\[MS-WMF\]](#) section 2.2.1.14), which specifies the output rectangle in logical coordinates.

**nDescription (4 bytes):** The number of Unicode characters in the optional description string that follows.

**Description (variable):** An optional, null-terminated Unicode string that describes this group of records.

Every EMR\_COMMENT\_BEGINGROUP record MUST be followed by an [EMR\\_COMMENT\\_ENDGROUP \(section 2.3.3.4.2\)](#) record in the metafile.

See section [2.3.3.4](#) for the specification of other EMR\_COMMENT\_PUBLIC record types.

#### 2.3.3.4.2 EMR\_COMMENT\_ENDGROUP Record

The Enhanced Metafile Format (EMF) EMR\_COMMENT\_ENDGROUP comment record identifies the end of a group of drawing records.

**Note** Fields that are not described below are specified in [EMR\\_COMMENT\\_PUBLIC Record Types \(section 2.3.3.4\)](#).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
DataSize																															
Identifier																															
CommentType																															

**CommentType (4 bytes):** A 32-bit unsigned integer that identifies the type of public comment record, from the [EmrComment enumeration \(section 2.1.10\)](#). It MUST be **EMR\_COMMENT\_ENDGROUP**, which is 0x00000003.

Every EMR\_COMMENT\_ENDGROUP record MUST be preceded by an [EMR\\_COMMENT\\_BEGINGROUP \(section 2.3.3.4.1\)](#) record in the metafile.

See section [2.3.3.4](#) for the specification of other EMR\_COMMENT\_PUBLIC record types.

#### 2.3.3.4.3 EMR\_COMMENT\_MULTIFORMATS Record

The Enhanced Metafile Format (EMF) EMR\_COMMENT\_MULTIFORMATS comment record allows multiple definitions of an image to be included in the metafile.

**Note** Fields that are not described below are specified in [EMR\\_COMMENT\\_PUBLIC Record Types \(section 2.3.3.4\)](#).



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
DataSize																															
Identifier																															
CommentType																															
OutputRect																															
...																															
...																															
...																															
CountFormats																															
aFormats (variable)																															
...																															
FormatData (variable)																															
...																															

**CommentType (4 bytes):** A 32-bit unsigned integer that identifies the type of public comment record, from the [EmrComment enumeration \(section 2.1.10\)](#). It MUST be **EMR\_COMMENT\_MULTIFORMATS**, which is 0x40000004.

**OutputRect (16 bytes):** A Windows Metafile Format (WMF) [RectL object](#) ([\[MS-WMF\]](#) section 2.2.1.14), which specifies the output rectangle in logical coordinates.

**CountFormats (4 bytes):** A 32-bit unsigned integer that specifies the number of formats contained in this record.

**aFormats (variable):** A **CountFormats** length array of [EmrFormat objects \(section 2.2.3\)](#), in order of preference.

**FormatData (variable):** A variable length array of bytes of data for each format. The size of the data for a given EmrFormat object is provided by the **DataSize** field in that EmrFormat object. Thus, the total size of this **FormatData** field is the sum of those **DataSize** values.

For example, an application can use this record type to specify an image in encapsulated PostScript as well as in EMF format. Subsequently, the PostScript version of the image MAY [≤38>](#) be rendered if that format is supported on the playback system.

See section [2.3.3.4](#) for the specification of other EMR\_COMMENT\_PUBLIC record types.

**2.3.3.4.4 EMR\_COMMENT\_WINDOWS\_METAFILE Record**

The Enhanced Metafile Format (EMF) EMR\_COMMENT\_WINDOWS\_METAFILE comment record specifies an image in an embedded Windows Metafile Format (WMF) or EMF metafile

**Note** Fields that are not described below are specified in [EMR\\_COMMENT\\_PUBLIC Record Types \(section 2.3.3.4\)](#).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
DataSize																															
Identifier																															
CommentType																															
Version																Reserved															
Checksum																															
Flags																															
WinMetaFileSize																															
MetaFileData (variable)																															
...																															

**CommentType (4 bytes):** A 32-bit unsigned integer that identifies the type of public comment record, from the [EmrComment enumeration \(section 2.1.10\)](#). It MUST be **EMR\_COMMENT\_WINDOWS\_METAFILE**, which is 0x00000001.

**Version (2 bytes):** A 16-bit unsigned integer that specifies the version of Windows Metafile Format (WMF) metafile. This value MUST be defined in the WMF [MetafileVersion enumeration](#) ([MS-WMF] section 2.1.21).

**Reserved (2 bytes):** This field is reserved, and the value MUST be 0x0000.

**Checksum (4 bytes):** A 32-bit unsigned integer that specifies the checksum for this record.

**Flags (4 bytes):** A 32-bit value that is reserved and MUST be 0x00000000.

**WinMetaFileSize (4 bytes):** A 32-bit unsigned integer that specifies the size of the WMF metafile data in bytes.

**MetaFileData (variable):** **WinMetaFileSize** bytes of WMF metafile data.

See section [2.3.3.4](#) for the specification of other EMR\_COMMENT\_PUBLIC record types.

### 2.3.4 Control Record Types

The Enhanced Metafile Format (EMF) control record types define the start and end of an EMF metafile and properties of the metafile.

The following are EMF control record types:

Name	Section	Description
EMR_EOF	<a href="#">2.3.4.1</a>	Indicates the end of the metafile and specifies a palette.
EMR_HEADER	<a href="#">2.3.4.2</a>	Indicates the start of the metafile and specifies properties of the device on which the metafile was created.

The generic structure of EMF control records is specified as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
dParm (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that defines the type of record. The control record types are listed below. See the table above for descriptions of these record types.

Name	Value
EMR_HEADER	0x00000001
EMR_EOF	0x0000000E

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size, in bytes, of the **dParm** field in the record.

**dParm (variable):** An array of 32-bit unsigned integers that specify parameters for the control record.

2.3.4.1 EMR\_EOF Record

The Enhanced Metafile Format EMR\_EOF record indicates the end of the metafile and specifies a palette.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
nPalEntries																															
offPalEntries																															
PaletteBuffer (variable)																															
...																															
SizeLast																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_EOF**. This MUST be 0x0000000E.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**nPalEntries (4 bytes):** A 32-bit unsigned integer that specifies the number of palette entries.

**offPalEntries (4 bytes):** A 32-bit unsigned integer that specifies the offset to the palette entries from the start of this record.

**PaletteBuffer (variable):** An optional buffer that contains palette data.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Padding1 (variable)																															
...																															
PaletteEntries (variable)																															
...																															
Padding2 (variable)																															
...																															

**Padding1 (variable):** Optional bytes of padding, because the **PaletteEntries** field is not required to immediately follow the **offPalEntries** field.

**PaletteEntries (variable):** An array of [LogPaletteEntry objects \(section 2.2.16\)](#) that specify the palette data.

**Padding2 (variable):** Optional bytes of padding, because the **PaletteEntries** field is not required to immediately precede the **SizeLast** field.

**SizeLast (4 bytes):** A 32-bit unsigned integer that MUST be the same as **Size** and MUST be the last field of the record. LogPaletteEntry objects, if they exist, MUST precede this field.

See section [2.3.4](#) for the specification of other Control record types.

### 2.3.4.2 EMR\_HEADER Record Types

The Enhanced Metafile Format (EMF) EMR\_HEADER record types indicate the start of the metafile and specify properties of the device on which the metafile was created. The values in the header record make it possible for EMF metafiles to be independent of any specific output device.

The following are EMF EMR\_HEADER record types:

Name	Section	Description
EmfMetafileHeader	<a href="#">2.3.4.2.1</a>	Defines the structure of the header used in the original version of EMF metafiles developed for Windows.
EmfMetafileHeaderExtension1	<a href="#">2.3.4.2.2</a>	Defines the structure of the header used in the first extension to the EMF metafile developed for Windows.
EmfMetafileHeaderExtension2	<a href="#">2.3.4.2.3</a>	Defines the structure of the header used in the second extension to the EMF metafile developed for Windows.

All EMF EMR\_HEADER records begin with a fixed-length **EMR\_HEADER** structure, specified below. In order to distinguish between the different EMR\_HEADER record types, the value of the **Size** field SHOULD be used.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Frame																															
...																															
...																															
...																															
Signature																															
Version																															
Bytes																															
Records																															
Handles																Reserved															
nDescription																															
offDescription																															
nPalEntries																															
Device																															

...
Millimeters
...

**Type (4 bytes):** A 32-bit unsigned integer that identifies this as an **EMR\_HEADER** structure. This MUST be 0x00000001.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the record size in bytes. This value SHOULD be used to determine the specific type of EMR\_HEADER record.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format (WMF) [RectL](#) object ([\[MS-WMF\]](#) section 2.2.1.14), which specifies the rectangular **inclusive-inclusive** bounds in device units of the smallest rectangle that can be drawn around the image stored in the metafile.

**Frame (16 bytes):** A 128-bit WMF **RectL** object that specifies the rectangular inclusive-inclusive dimensions, in .01 millimeter units, of a rectangle that surrounds the image stored in the metafile.

**Signature (4 bytes):** A 32-bit unsigned integer that specifies the record signature, which MUST be **ENHMETA\_SIGNATURE**, specified in the [FormatSignature enumeration \(section 2.1.14\)](#).

**Version (4 bytes):** A 32-bit unsigned integer that specifies the version of Windows interoperability that is required for the EMF metafile. This value MUST be defined in the [MetafileVersion enumeration \(section 2.1.22\)](#).

**Bytes (4 bytes):** A 32-bit unsigned integer that specifies the size of the metafile in bytes.

**Records (4 bytes):** A 32-bit unsigned integer that specifies the number of records in the metafile.

**Handles (2 bytes):** A 16-bit unsigned integer that specifies the number of [EMF Object Table](#) entries that will need to be defined during the processing of the metafile. These entries correspond to graphics objects that are used in drawing commands. The entries are indexed; the indexes are used to refer indirectly to the graphics objects. Index 0 is reserved for references to the metafile itself.

**Reserved (2 bytes):** A 16-bit unsigned integer that is not used and MUST be 0x0000.

**nDescription (4 bytes):** A 32-bit unsigned integer that specifies the number of characters in the array that contains the description of the metafile's contents. This is zero if there is no description string.

**offDescription (4 bytes):** A 32-bit unsigned integer that specifies the offset from the beginning of this record to the array that contains the description of the metafile's contents.

**nPalEntries (4 bytes):** A 32-bit unsigned integer that specifies the number of entries in the metafile palette. The palette is located in the [EMR\\_EOF](#) record.

**Device (8 bytes):** A 64-bit WMF [SizeL](#) object ([\[MS-WMF\]](#) section 2.2.1.16), that specifies the size of the reference device in pixels.

**Millimeters (8 bytes):** A 64-bit WMF SizeL object that specifies the size of the reference device in millimeters.

See section [2.3.4](#) for the specification of other Control record types.

**2.3.4.2.1 EmfMetafileHeader Record**

The Enhanced Metafile Format (EMF) EmfMetafileHeader record defines the structure of the header used in the original version of EMF metafiles developed by Microsoft.

The **EmfHeader** field **MUST** always be present at the start of the EmfMetafileHeader record. Following the **EmfHeader** field, an optional **EmfDescription** field, which contains a string, **MAY** be present.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EmfHeader																															
...																															
...																															
...																															
...																															
...																															
...																															
(EmfHeader cont'd for 14 rows)																															
EmfDescription (variable)																															
...																															

**EmfHeader (88 bytes):** The **EmfHeader** field contains information about the content and structure of the metafile. It is a record of type [EMR\\_HEADER](#), as specified in section [2.3.4.2](#).

**EmfDescription (variable):** The optional **EmfDescription** field specifies a description string of arbitrary length and content. Its actual length and position in the metafile can be determined from the description length and offset fields, respectively, of the **EmfHeader** field; if either the description length or offset is 0, no description string is present.

See section [2.3.4.2](#) for the specification of other Header record types.



### 2.3.4.2.2 EmfMetafileHeaderExtension1 Record

The Enhanced Metafile Format EmfMetafileHeaderExtension1 record defines the structure of the header used in the first extension to the EMF metafile developed for Windows.

The **EmfHeader** field, defined below, MUST always be present at the start of the EmfMetafileHeaderExtension1 record. It MAY [≤39>](#) be followed by an **EmfHeaderExtension1** field, defined below, which contains a [HeaderExtension1](#) object, specified in section [2.2.8](#). If the **EmfHeaderExtension1** field is not present, the metafile header is simply the "original" type, specified in section [2.3.4.2.1](#). The **EmfPixelFormat** field SHOULD be present only if the **EmfHeaderExtension1** field is also present.

Following the **EmfHeaderExtension1** field, the remaining fields MAY be present in any order.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EmfHeader																															
...																															
...																															
...																															
...																															
...																															
...																															
(EmfHeader cont'd for 14 rows)																															
EmfHeaderExtension1 (optional)																															
...																															
...																															
EmfDescription (variable)																															
...																															

EmfPixelFormat (optional)
...
...
...
...
...
...
...
(EmfPixelFormat (optional) cont'd for 2 rows)

**EmfHeader (88 bytes):** The **EmfHeader** field contains information about the content and structure of the metafile. It is a record of type [EMR\\_HEADER](#), as specified in section [2.3.4.2](#).

**EmfHeaderExtension1 (12 bytes):** The optional **EmfHeaderExtension1** field contains additional information about the remaining fields in the header record. It is a [HeaderExtension1](#) object.

**EmfDescription (variable):** The optional **EmfDescription** field specifies a description string of arbitrary length and content. Its actual length and position in the metafile can be determined from the description length and offset fields, respectively, of the **EmfHeader** field; if either the description length or offset is 0, no description string is present.

**EmfPixelFormat (40 bytes):** The **EmfPixelFormat** field specifies the last pixel format that was defined when the metafile was recorded. It is of type [PixelFormatDescriptor](#) (section [2.2.20](#)).

**Note** No single structure definition can accurately represent every possible combination of optional fields. Therefore, the implementer is responsible for writing software that determines which fields are actually present in a given metafile, and for unmarshaling the contents of each field appropriately.

See section [2.3.4.2](#) for the specification of other Header record types.

#### 2.3.4.2.3 EmfMetafileHeaderExtension2 Record

The Enhanced Metafile Format **EmfMetafileHeaderExtension2** record defines the structure of the header used in the second extension to the EMF metafile developed for Windows.

The **EmfHeader** field, defined below, **MUST** always be present at the start of the **EmfMetafileHeaderExtension2** record. It **MAY** [<40>](#) be followed by an **EmfHeaderExtension1** field, defined below, which contains a [HeaderExtension1](#) object, specified in section [2.2.8](#). If the

**EmfHeaderExtension1** object is not present, the metafile header is simply the "original" type, specified in section [2.3.4.2.1](#). The **EmfPixelFormat** field SHOULD be present only if the **EmfHeaderExtension1** object is also present.

If the **EmfHeaderExtension1** field is present, it MAY be followed by an **EmfHeaderExtension2** field, defined below, which contains a [HeaderExtension2](#) object, specified in section [2.2.9](#). If the **EmfHeaderExtension2** object is not present, the metafile header is simply the "extension 1" type, specified in section [2.3.4.2.2](#).

Following the **EmfHeaderExtension2** field, the remaining fields MAY be present in any order.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EmfHeader																															
...																															
...																															
...																															
...																															
...																															
...																															
(EmfHeader cont'd for 14 rows)																															
EmfHeaderExtension1 (optional)																															
...																															
...																															
EmfHeaderExtension2 (optional)																															
...																															
EmfDescription (variable)																															

...
EmfPixelFormat (optional)
...
...
...
...
...
...
...
...
(EmfPixelFormat (optional) cont'd for 2 rows)

**EmfHeader (88 bytes):** The **EmfHeader** field contains information about the content and structure of the metafile. It is a record of type [EMR\\_HEADER](#), as specified in section [2.3.4.2](#).

**EmfHeaderExtension1 (12 bytes):** The **EmfHeaderExtension1** field contains additional information about the remaining fields in the header record. It is a [HeaderExtension1](#) object.

**EmfHeaderExtension2 (8 bytes):** The **EmfHeaderExtension2** field contains additional information about the image in the metafile. It is a [HeaderExtension2](#) object.

**EmfDescription (variable):** The optional **EmfDescription** field specifies a description string of arbitrary length and content. Its actual length and position in the metafile can be determined from the description length and offset fields, respectively, of the **EmfHeader** field; if either the description length or offset is 0, no description string is present.

**EmfPixelFormat (40 bytes):** The **EmfPixelFormat** field specifies the last pixel format that was defined when the metafile was recorded. It is type [PixelFormatDescriptor \(section 2.2.20\)](#).

**Note** No single structure definition can accurately represent every possible combination of optional fields. Therefore, the implementer is responsible for writing software that determines which fields are actually present in a given metafile, and for unmarshaling the contents of each field appropriately.

See section [2.3.4.2](#) for the specification of other Header record types.

### 2.3.5 Drawing Record Types

The Enhanced Metafile Format (EMF) drawing record types perform graphics drawing.

The following are EMF drawing record types:

Name	Section	Description
EMR_ANGLEARC	<a href="#">2.3.5.1</a>	Draws a line segment of an arc.
EMR_ARC	<a href="#">2.3.5.2</a>	Draws an elliptical arc.
EMR_ARCTO	<a href="#">2.3.5.3</a>	Draws an elliptical arc, resetting the current drawing position to the end point of the arc.
EMR_CHORD	<a href="#">2.3.5.4</a>	Draws a chord, which is a region bounded by the intersection of an ellipse and a line segment, called a secant.
EMR_ELLIPSE	<a href="#">2.3.5.5</a>	Draws an ellipse.
EMR_EXTFLOODFILL	<a href="#">2.3.5.1</a>	Draws a line segment of an arc.
EMR_EXTTEXTOUTA	<a href="#">2.3.5.7</a>	Draws ASCII text with the currently selected font, background color, and text color.
EMR_EXTTEXTOUTW	<a href="#">2.3.5.8</a>	Draws Unicode text with the currently selected font, background color, and text color.
EMR_FILLPATH	<a href="#">2.3.5.9</a>	Closes any open figures in the current path and fills the path's interior with the current brush and polygon-filling mode.
EMR_FILLRGN	<a href="#">2.3.5.10</a>	Fills the specified region with the specified brush.
EMR_FORCEUFIMAPPING	<a href="#">2.3.11.2</a>	Forces the font mapper to match fonts based on their <b>UniversalFontId</b> in preference to their <a href="#">LogFont (section 2.2.11)</a> information.
EMR_FRAMERGN	<a href="#">2.3.5.11</a>	Draws a border around the specified region with the specified brush.
EMR_GRADIENTFILL	<a href="#">2.3.5.12</a>	Fills the specified rectangle and triangle structures.
EMR_LINETO	<a href="#">2.3.5.13</a>	Draws a line from the current position up to, but not including, the specified point. This record resets the current position to that point.
EMR_PAINTRGN	<a href="#">2.3.5.14</a>	Paints the specified region with the current brush.
EMR_PIE	<a href="#">2.3.5.15</a>	Draws a pie-shaped wedge bounded by the intersection of an ellipse and two radials.
EMR_POLYBEZIER	<a href="#">2.3.5.16</a>	Draws one or more Bezier curves. The cubic Bezier curves are defined with the endpoints and control points specified in this record.
EMR_POLYBEZIER16	<a href="#">2.3.5.17</a>	Draws one or more Bezier curves with the current pen.
EMR_POLYBEZIERTO	<a href="#">2.3.5.18</a>	Draws one or more Bezier curves based on the current position.
EMR_POLYBEZIERTO16	<a href="#">2.3.5.19</a>	Draws one or more Bezier curves based on the current position.
EMR_POLYDRAW	<a href="#">2.3.5.20</a>	Draws a set of line segments and Bezier curves.
EMR_POLYDRAW16	<a href="#">2.3.5.21</a>	Draws a set of line segments and Bezier curves.
EMR_POLYGON	<a href="#">2.3.5.22</a>	Draws a polygon consisting of two or more vertices connected

Name	Section	Description
		by straight lines.
EMR_POLYGON16	<a href="#">2.3.5.23</a>	Draws a polygon consisting of two or more vertices connected by straight lines.
EMR_POLYLINE	<a href="#">2.3.5.24</a>	Draws a series of line segments by connecting the points in the specified array.
EMR_POLYLINE16	<a href="#">2.3.5.25</a>	Draws a series of line segments by connecting the points in the specified array.
EMR_POLYLINETO	<a href="#">2.3.5.26</a>	Draws one or more straight lines based upon the current position.
EMR_POLYLINETO16	<a href="#">2.3.5.27</a>	Draws one or more straight lines based upon the current position.
EMR_POLYPOLYGON	<a href="#">2.3.5.28</a>	Paints a series of closed polygons. Each polygon is outlined with the current pen and filled with the current brush and polygon fill mode.
EMR_POLYPOLYGON16	<a href="#">2.3.5.29</a>	Paints a series of closed polygons. Each polygon is outlined with the current pen and filled with the current brush and polygon fill mode.
EMR_POLYPOLYLINE	<a href="#">2.3.5.30</a>	Draws multiple series of connected line segments.
EMR_POLYPOLYLINE16	<a href="#">2.3.5.31</a>	Draws multiple series of connected line segments.
EMR_POLYTEXTOUTA	<a href="#">2.3.5.32</a>	Draws several ASCII strings with the currently selected font, background color, and text color.
EMR_POLYTEXTOUTW	<a href="#">2.3.5.33</a>	Draws several Unicode strings with the currently selected font, background color, and text color.
EMR_RECTANGLE	<a href="#">2.3.5.34</a>	Draws a rectangle. The rectangle is outlined with the current pen and filled with the current brush.
EMR_ROUNDRECT	<a href="#">2.3.5.35</a>	Draws a rectangle with rounded corners.
EMR_SETPIXELV	<a href="#">2.3.5.36</a>	Defines the color of the pixel at the specified logical coordinates.
EMR_SMALLTEXTOUT	<a href="#">2.3.5.37</a>	Outputs a string.
EMR_STROKEANDFILLPATH	<a href="#">2.3.5.38</a>	Closes any open figures in a path, draws the outline of the path with the current pen, and fills its interior with the current brush.
EMR_STROKEPATH	<a href="#">2.3.5.39</a>	Draws the specified path with the current pen.

The generic structure of EMF drawing records is specified as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
dParm (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that defines the type of record. The drawing record types are listed below. See the table above for descriptions of these records.

Name	Value
EMR_POLYBEZIER	0x00000002
EMR_POLYGON	0x00000003
EMR_POLYBEZIERTO	0x00000005
EMR_POLYLINETO	0x00000006
EMR_POLYPOLYLINE	0x00000007
EMR_POLYPOLYGON	0x00000008
EMR_SETPIXELV	0x0000000F
EMR_ANGLEARC	0x00000029
EMR_ELLIPSE	0x0000002A
EMR_RECTANGLE	0x0000002B
EMR_ROUNDRECT	0x0000002C
EMR_ARC	0x0000002D
EMR_CHORD	0x0000002E
EMR_PIE	0x0000002F
EMR_EXTFLOODFILL	0x00000035
EMR_LINETO	0x00000036
EMR_ARCTO	0x00000037
EMR_POLYDRAW	0x00000038
EMR_FILLPATH	0x0000003E
EMR_STROKEANDFILLPATH	0x0000003F

Name	Value
EMR_STROKEPATH	0x00000040
EMR_FILLRGN	0x00000047
EMR_FRAMERGN	0x00000048
EMR_PAINTRGN	0x0000004A
EMR_EXTTEXTOUTA	0x00000053
EMR_EXTTEXTOUTW	0x00000054
EMR_POLYBEZIER16	0x00000055
EMR_POLYGON16	0x00000056
EMR_POLYLINE16	0x00000057
EMR_POLYBEZIERTO16	0x00000058
EMR_POLYLINETO16	0x00000059
EMR_POLYPOLYLINE16	0x0000005A
EMR_POLYPOLYGON16	0x0000005B
EMR_POLYDRAW16	0x0000005C
EMR_POLYTEXTOUTA	0x00000060
EMR_POLYTEXTOUTW	0x00000061
EMR_SMALLTEXTOUT	0x0000006C
EMR_GRADIENTFILL	0x00000076

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size, in bytes, of the **dParm** field in the record.

**dParm (variable):** An array of 32-bit unsigned integers that specify parameters for the drawing record.

### 2.3.5.1 EMR\_ANGLEARC Record

The Enhanced Metafile Format EMR\_ANGLEARC record specifies a line segment of an arc. The line segment is drawn from the current position to the beginning of the arc. The arc is drawn along the perimeter of a circle with the given radius and center. The length of the arc is defined by the given start and sweep angles.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Center																															
...																															
Radius																															
StartAngle																															
SweepAngle																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_ANGLEARC**. This MUST be 0x00000029.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Center (8 bytes):** A 64-bit Windows Metafile Format **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies the logical coordinates of the circle's center.

**Radius (4 bytes):** A 32-bit unsigned integer that specifies the circle's radius in logical units.

**StartAngle (4 bytes):** A 32-bit float that specifies the arc's start angle in degrees.

**SweepAngle (4 bytes):** A 32-bit float that specifies the arc's sweep angle in degrees.

The arc is drawn by recording an imaginary circle around the specified center point with the specified radius. The starting point of the arc is determined by measuring counterclockwise from the x-axis of the circle by the number of degrees in the start angle. The ending point is similarly located by measuring counterclockwise from the starting point by the number of degrees in the sweep angle.

If the sweep angle is greater than 360 degrees, the arc is swept multiple times.

This record specifies lines by using the current pen. The figure is not filled.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.2 EMR\_ARC Record

The Enhanced Metafile Format EMR\_ARC record specifies an elliptical arc.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Box																															
...																															
...																															
...																															
Start																															
...																															
End																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_ARC**. This MUST be 0x0000002D.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Box (16 bytes):** A 128-bit WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the inclusive-inclusive bounding rectangle.

**Start (8 bytes):** A 64-bit WMF **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, which specifies the coordinates, in logical units, of the ending point of the radial line defining the starting point of the arc.

**End (8 bytes):** A 64-bit WMF **PointL** object that specifies the coordinates, in logical units, of the ending point of the radial line defining the ending point of the arc.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.3 EMR\_ARCTO Record

The Enhanced Metafile Format EMR\_ARCTO record specifies an elliptical arc. It resets the current position to the end point of the arc.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Box																															
...																															
...																															
...																															
Start																															
...																															
End																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_ARCTO**. This MUST be 0x00000037.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Box (16 bytes):** A 128-bits [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle.

**Start (8 bytes):** A 64-bit WMF **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, which specifies the coordinates of first radial ending point in logical units.

**End (8 bytes):** A 64-bit WMF **PointL** object that specifies the coordinates of second radial ending point in logical units.

See section [2.3.5](#) for the specification of other Drawing record types.

#### 2.3.5.4 EMR\_CHORD Record

The Enhanced Metafile Format EMR\_CHORD record specifies a chord, which is a region bounded by the intersection of an ellipse and a line segment, called a secant. The chord is outlined by using the current pen and filled by using the current brush.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Box																															
...																															
...																															
...																															
Start																															
...																															
End																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_CHORD**. This MUST be 0x0000002E.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Box (16 bytes):** A 128-bit [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the inclusive-inclusive bounding rectangle.

**Start (8 bytes):** A 64-bit WMF **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, which specifies the logical coordinates of the endpoint of the radial defining the beginning of the chord.

**End (8 bytes):** A 64-bit WMF **PointL** object that specifies the logical coordinates of the endpoint of the radial defining the end of the chord.

The curve of the chord is defined by an ellipse that fits the specified bounding rectangle. The curve begins at the point where the ellipse intersects the first radial and extends counterclockwise to the point where the ellipse intersects the second radial. The chord is closed by drawing a line from the intersection of the first radial and the curve to the intersection of the second radial and the curve.

If the starting point and ending point of the curve are the same, a complete ellipse is drawn.

The current position is neither used nor updated by processing this record.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.5 EMR\_ELLIPSE Record

The Enhanced Metafile Format EMR\_ELLIPSE record specifies an ellipse. The center of the ellipse is the center of the specified bounding rectangle. The ellipse is outlined by using the current pen and is filled by using the current brush.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Box																															
...																															
...																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_ELLIPSE**. This MUST be 0x0000002A.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Box (16 bytes):** A 128-bit [WMF RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the inclusive-inclusive bounding rectangle.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.6 EMR\_EXTFLOODFILL Record

The Enhanced Metafile Format EMR\_EXTFLOODFILL record fills an area of the display surface with the current brush.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Start																															
...																															
Color																															
FloodFillMode																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_EXTFLOODFILL**. This MUST be 0x00000035.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Start (8 bytes):** A 64-bit Windows Metafile Format **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies the coordinates, in logical units, where filling begins.

**Color (4 bytes):** A 32-bit WMF **ColorRef** object, specified in [\[MS-WMF\]](#) section 2.2.1.7, that specifies the area to fill.

**FloodFillMode (4 bytes):** A 32-bit unsigned integer that specifies the type of fill operation to be performed. The value MUST be in the [FloodFill \(section 2.1.13\)](#) enumeration.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.7 EMR\_EXTTEXTOUTA Record

The Enhanced Metafile Format EMR\_EXTTEXTOUTA record specifies text using the currently selected font, background color, and text color.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
iGraphicsMode																															
exScale																															
eyScale																															
mrtext (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_EXTTEXTOUTA**. This MUST be 0x00000053.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle in device units. It MAY be used for clipping, opaquing, or both.

**iGraphicsMode (4 bytes):** A 32-bit unsigned integer that specifies the current graphics mode. Graphics modes are specified in section [2.1.16](#).

**exScale (4 bytes):** A 32-bit float that specifies the X scale from page units to .01mm units if graphics mode is **GM\_COMPATIBLE**.

**eyScale (4 bytes):** A 32-bit float that specifies the Y scale from page units to .01mm units if graphics mode is **GM\_COMPATIBLE**.

**mrtext (variable):** An [EmrText \(section 2.2.4\)](#) object that specifies the properties of the string to be printed, and where to find the output string and spacing values.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.8 EMR\_EXTTEXTOUTW Record

The Enhanced Metafile Format EMR\_EXTTEXTOUTW record specifies text using the currently selected font, background color, and text color. An optional rectangle MAY be used for clipping, opaquing, or both.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
iGraphicsMode																															
exScale																															
eyScale																															
mrttext (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_EXTTEXTOUTW**. This MUST be 0x00000054.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [WMF RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle in device units.

**iGraphicsMode (4 bytes):** A 32-bit unsigned integer that specifies the current graphics mode. Graphics modes are specified in section [2.1.16](#).

**exScale (4 bytes):** A 32-bit float that specifies the X scale from page units to .01mm units if the graphics mode is **GM\_COMPATIBLE**. See [GraphicsMode \(section 2.1.16\)](#) enumeration for details.

**eyScale (4 bytes):** A 32-bit float that specifies the Y scale from page units to .01mm units if graphics mode is **GM\_COMPATIBLE**.



**mrtext (variable):** An [EmrText \(section 2.2.4\)](#) object that specifies the properties of the string to be printed, and where to find the output string and spacing values.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.9 EMR\_FILLPATH Record**

The Enhanced Metafile Format EMR\_FILLPATH record closes any open figures in the current path and fills the path's interior by using the current brush and polygon-filling mode.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_FILLPATH**. This MUST be 0x0000003E.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [WMF RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies bounding rectangle in device units.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.10 EMR\_FILLRGN Record**

The Enhanced Metafile Format EMR\_FILLRGN record fills the specified region by using the specified brush.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
RgnDataSize																															
ihBrush																															
RgnData (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_FILLRGN**. This MUST be 0x00000047.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle.

**RgnDataSize (4 bytes):** A 32-bit unsigned integer that specifies the size of region data in bytes.

**ihBrush (4 bytes):** A 32-bit unsigned integer that specifies the brush [EMF Object Table](#) index to fill the region.

**RgnData (variable):** A **RgnDataSize** length array of bytes that contains a [RegionData](#) (section 2.2.21) object.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.11 EMR\_FRAMERGN Record

The Enhanced Metafile Format EMR\_FRAMERGN record draws a border around the specified region using the specified brush.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
RgnDataSize																															
ihBrush																															
Width																															
Height																															
RgnData (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_FRAMERGN**. This MUST be 0x00000048.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** 128-bit [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle.

**RgnDataSize (4 bytes):** A 32-bit unsigned integer that specifies the size of region data in bytes.

**ihBrush (4 bytes):** A 32-bit unsigned integer that specifies the brush [EMF Object Table](#) index.

**Width (4 bytes):** A 32-bit signed integer that specifies the width of the vertical brush stroke in logical units.

**Height (4 bytes):** A 32-bit signed integer that specifies the height of the horizontal brush stroke in logical units.

**RgnData (variable):** A **RgnDataSize** length array of bytes that specifies a [RegionData](#) object in logical units.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.12 EMR\_GRADIENTFILL Record**

The Enhanced Metafile Format EMR\_GRADIENTFILL record fills the specified rectangle and triangle structures.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
nVer																															
nTri																															
ulMode																															
VertexData (variable)																															
...																															

- Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as EMR\_GRADIENTFILL. This MUST be 0x00000076.
- Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.
- Bounds (16 bytes):** A 128-bit [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the inclusive-inclusive bounds in device units.
- nVer (4 bytes):** A 32-bit unsigned integer that specifies the number of vertexes.
- nTri (4 bytes):** A 32-bit unsigned integer that specifies the number of rectangles or triangles.

**ulMode (4 bytes):** A 32-bit unsigned integer that specifies the gradient fill mode. The value MUST be in the [GradientFillMode Enumeration \(section 2.1.15\)](#).

**VertexData (variable):** Vertices defining rectangles or triangles, and the colors to fill them. Each vertex MAY have a different color defined for it, which means that the processing of this record MAY cause the figures to be filled with gradients of color from one vertex to another.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
VertexObjects (variable)																															
...																															
VertexIndexes (variable)																															
...																															

**VertexObjects (variable):** An array of **nVer** [TriVertex \(section 2.2.23\)](#) objects. Each object specifies the position and color of a vertex for either a rectangle or triangle, depending on the value of the **ulMode** field.

**VertexIndexes (variable):** An array of **nTri** [GradientRectangle \(section 2.2.6\)](#) or [GradientTriangle \(section 2.2.7\)](#) objects. Each object specifies indexes into the **VertexObjects** array, and thus the vertices for either a rectangle or triangle, depending on the value of the **ulMode** field.

The type of object in this field is obvious if one knows the size of an array element in bytes, and that can be computed from other fields in this record, according to the following equation:

$$\text{ElementSize} = (\text{Size} - (36 + (\text{nVer} * 16))) / \text{nTri}$$

This equation simply divides the size of the **VertexIndexes** field by the number of elements to compute the size of a single element in bytes. If the result is 8, the object type is **GradientRectangle**; if the result is 12, the object type is **GradientTriangle**.

An **EMR\_GRADIENTFILL** record that specifies the three vertices of a triangle SHOULD fill the figure with smooth gradients of colors.

An **EMR\_GRADIENTFILL** record that specifies the upper-left and lower-right vertices of a rectangle SHOULD fill the figure with smooth gradients of color. There are two shading modes in the **GradientFillMode Enumeration** that can be used when drawing a rectangle. In horizontal mode, the rectangle is shaded from left-to-right. In vertical mode, the rectangle is shaded from top-to-bottom.

**Note** An **EMR\_GRADIENTFILL** record SHOULD NOT use the **Alpha** fields from the **TriVertex** objects, which define transparency, unless it is immediately followed by an [EMR\\_ALPHABLEND \(section 2.3.1.1\)](#) record with the alpha value of each vertex.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.13 EMR\_LINETO Record**

The Enhanced Metafile Format EMR\_LINETO record specifies a line from the current position up to, but not including, the specified point. It resets the current position to the specified point.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Point																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_LINETO**. This MUST be 0x00000036.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Point (8 bytes):** A 64-bit Windows Metafile Format **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies the coordinates of the line's ending point.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.14 EMR\_PAINTRGN Record**

The Enhanced Metafile Format EMR\_PAINTRGN record paints the specified region by using the brush currently selected into the playback device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
RgnDataSize																															
RgnData (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_PAINTRGN**. This MUST be 0x0000004A.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle.

**RgnDataSize (4 bytes):** A 32-bit unsigned integer that specifies the size of region data in bytes.

**RgnData (variable):** A **RgnDataSize** length array of bytes that specifies a [RegionData \(section 2.2.21\)](#) object in logical units.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.15 EMR\_PIE Record

The Enhanced Metafile Format EMR\_PIE record specifies a pie-shaped wedge bounded by the intersection of an ellipse and two radials. The pie is outlined by using the current pen and filled by using the current brush.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Box																															
...																															
...																															
...																															
Start																															
...																															
End																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_PIE**. This MUST be 0x0000002F.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Box (16 bytes):** A 128-bits [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the inclusive-inclusive bounding rectangle.

**Start (8 bytes):** A 64-bits WMF **PointL** objects, specified in [\[MS-WMF\]](#) section 2.2.1.11, which specifies the coordinates, in logical units, of the endpoint of the first radial.

**End (8 bytes):** A 64-bits **PointL** object that specifies the coordinates, in logical units, of the endpoint of the second radial.

The curve of the pie is defined by an ellipse that fits the specified bounding rectangle. The curve begins at the point where the ellipse intersects the first radial and extends counterclockwise to the point where the ellipse intersects the second radial.

The current position is neither used nor updated by this record.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.16 EMR\_POLYBEZIER Record

The Enhanced Metafile Format (EMF) EMR\_POLYBEZIER record specifies one or more Bezier curves.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYBEZIER**. This MUST be 0x00000002.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format (WMF) **RectL** object ([\[MS-WMF\]](#) section 2.2.1.14) that specifies the bounding rectangle in device units.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the number of points in the **aPoints** array. This value MUST be one more than three times the number of curves to be drawn, because each Bezier curve requires two control points and an endpoint, and the initial curve requires an additional starting point.

Line width	Device supports wideline	Maximum points allowed
1	n/a	16K
> 1	yes	16K
> 1	no	1360

Any extra points MUST be ignored.

**aPoints (variable):** A **Count** length array of WMF **PointL** objects ([\[MS-WMF\]](#) section 2.2.1.11) that specify the endpoints and control points of the Bezier curves in logical units.

Cubic Bezier curves are defined using the endpoints and control points specified by the **aPoints** field. The first curve is drawn from the first point to the fourth point, using the second and third

points as control points. Each subsequent curve in the sequence needs exactly three more points: the ending point of the previous curve is used as the starting point, the next two points in the sequence are control points, and the third is the ending point.

The cubic Bezier curves SHOULD be drawn using the current pen.

See section [2.3.5](#) for the specification of other drawing record types.

**2.3.5.17 EMR\_POLYBEZIER16 Record**

The Enhanced Metafile Format EMR\_POLYBEZIER16 record specifies one or more Bezier curves. The curves are drawn using the current pen.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															

- Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYBEZIER16**. This MUST be 0x00000055.
- Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.
- Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.
- Count (4 bytes):** A 32-bit unsigned integer that specifies the total number of points. This value MUST be one more than three times the number of curves to be drawn, because each Bezier curve requires two control points and an endpoint, and the initial curve requires an additional starting point
- aPoints (variable):** A **Count** length array of WMF **Points** objects, specified in [\[MS-WMF\]](#) section 2.2.1.12, that specifies the array of points.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.18 EMR\_POLYBEZIERTO Record

The Enhanced Metafile Format (EMF) EMR\_POLYBEZIERTO record specifies one or more Bezier curves based upon the current position.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYBEZIERTO**. This MUST be 0x00000005.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format (WMF) **RectL** object ([\[MS-WMF\]](#) section 2.2.1.14) that specifies the bounding rectangle in device units.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the number of points in the **aPoints** array. The first curve MUST be drawn from the current position to the third point by using the first two points as control points. For each subsequent curve, exactly three more points MUST be specified, and the ending point of the previous curve MUST be used as the starting point for the next.

Line width	Device supports wideline	Maximum points allowed
1	n/a	16K
> 1	yes	16K
> 1	no	1360

Any extra points MUST be ignored.

**aPoints (variable):** A **Count** length array of WMF **PointL** objects ([\[MS-WMF\]](#) section 2.2.1.11) that specify the endpoints and control points of the Bezier curves in logical units.

The Bezier curves SHOULD be drawn using the current pen.

See section [2.3.5](#) for the specification of other drawing record types.

### 2.3.5.19 EMR\_POLYBEZIERTO16 Record

The Enhanced Metafile Format (EMF) EMR\_POLYBEZIERTO16 record specifies one or more Bezier curves based on the current position.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYBEZIERTO16**. This MUST be 0x00000058.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the total number of points. The first curve is drawn from the current position to the third point by using the first two points as control points. For each subsequent curve, three more points MUST be specified, and the ending point of the previous curve MUST be used as the starting point for the next.

**aPoints (variable):** A **Count** length array of WMF **PointS** objects, specified in [\[MS-WMF\]](#) section 2.2.1.12, that specify the array of points.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.20 EMR\_POLYDRAW Record**

The Enhanced Metafile Format EMR\_POLYDRAW record specifies a set of line segments and Bezier curves.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															
abTypes (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYDRAW**. This MUST be 0x00000038.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the number of points.

**aPoints (variable):** A **Count** length array of WMF **PointL** objects, specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies the array of points in logical units.

**abTypes (variable):** A **Count** length array of bytes that specifies the array of values that specifies how each point in the **aPoints** array is used. This value MUST be in the [Point \(section 2.1.26\)](#) enumeration.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.21 EMR\_POLYDRAW16 Record**

The Enhanced Metafile Format EMR\_POLYDRAW16 record specifies a set of line segments and Bezier curves.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															
abTypes (variable)																															
...																															

- Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYDRAW16**. This MUST be 0x0000005C.
- Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.
- Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.
- Count (4 bytes):** A 32-bit unsigned integer that specifies the number of points.
- aPoints (variable):** A **Count** length array of WMF **Points** objects, specified in [\[MS-WMF\]](#) section 2.2.1.12, that specifies the array of points.

**abTypes (variable):** A **Count** length array of bytes that specifies the point types. This value MUST be in the [Point \(section 2.1.26\)](#) enumeration.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.22 EMR\_POLYGON Record**

The Enhanced Metafile Format (EMF) EMR\_POLYGON record specifies a polygon consisting of two or more vertices connected by straight lines.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYGON**. This MUST be 0x00000003.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format (WMF) **RectL** object ([\[MS-WMF\]](#) section 2.2.1.14) that specifies the bounding rectangle in device units.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the number of points in the **aPoints** array.

Line width	Device supports wideline	Maximum points allowed
1	n/a	16K
> 1	yes	16K
> 1	no	1360

Any extra points MUST be ignored.

**aPoints (variable):** A **Count** length array of WMF **PointL** objects ([\[MS-WMF\]](#) section 2.2.1.11) that specify the vertices of the polygon in logical units.

The polygon SHOULD be outlined using the current pen and filled using the current brush and polygon fill mode. The polygon SHOULD be closed automatically by drawing a line from the last vertex to the first.

See section [2.3.5](#) for the specification of other drawing record types.

### 2.3.5.23 EMR\_POLYGON16 Record

The Enhanced Metafile Format (EMF) EMR\_POLYGON16 record specifies a polygon consisting of two or more vertices connected by straight lines. The polygon is outlined by using the current pen and filled by using the current brush and polygon fill mode. The polygon is closed automatically by drawing a line from the last vertex to the first.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYGON16**. This MUST be 0x00000056.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the total number of points.



**aPoints (variable):** A **Count** length array of WMF **Points** objects, specified in [\[MS-WMF\]](#) section 2.2.1.12, that specifies the array of points.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.24 EMR\_POLYLINE Record

The Enhanced Metafile Format (EMF) EMR\_POLYLINE record specifies a series of line segments by connecting the points in the specified array.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYLINE**. This MUST be 0x00000004.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format (WMF) **RectL** object ([\[MS-WMF\]](#) section 2.2.1.14) that specifies the bounding rectangle in device units.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the number of points in the **aPoints** array.

Line width	Device supports wideline	Maximum points allowed
1	n/a	16K
> 1	yes	16K
> 1	no	1360

Any extra points MUST be ignored.

**aPoints (variable):** A **Count** length array of WMF **PointL** objects ([\[MS-WMF\]](#) section 2.2.1.11) that specify the point data in logical units.

The line segments SHOULD be drawn using the current pen.

See section [2.3.5](#) for the specification of other drawing record types.

### 2.3.5.25 EMR\_POLYLINE16 Record

The Enhanced Metafile Format EMR\_POLYLINE16 record specifies a series of line segments by connecting the points in the specified array.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYLINE16**. This MUST be 0x00000057.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the total number of points.

**aPoints (variable):** A **Count** length array of WMF **PointS** objects, specified in [\[MS-WMF\]](#) section 2.2.1.12, that specifies the array of points.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.26 EMR\_POLYLINETO Record

The Enhanced Metafile Format (EMF) EMR\_POLYLINETO record specifies one or more straight lines based upon the current position.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYLINETO**. This MUST be 0x00000006.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format (WMF) **RectL** object ([\[MS-WMF\]](#) section 2.2.1.14) that specifies the bounding rectangle in device units.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the number of points in the **aPoints** array.

Line width	Device supports wideline	Maximum points allowed
1	n/a	16K
> 1	yes	16K
> 1	no	1360

Any extra points MUST be ignored.

**aPoints (variable):** A **Count** length array of WMF **PointL** objects ([\[MS-WMF\]](#) section 2.2.1.11) that specify the point data in logical units.

A line SHOULD be drawn from the current position to the first point specified by the **aPoints** field using the current pen. Each additional line SHOULD be drawn from the ending point of the previous line to the next point specified by **aPoints**.

See section [2.3.5](#) for the specification of other drawing record types.

**2.3.5.27 EMR\_POLYLINE16 Record**

The Enhanced Metafile Format **EMR\_POLYLINE16** record specifies one or more straight lines based upon the current position. A line is drawn from the current position to the first point specified by the **aPoints** field by using the current pen. For each additional line, drawing is performed from the ending point of the previous line to the next point specified by **aPoints**.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
Count																															
aPoints (variable)																															
...																															

- Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYLINE16**. This MUST be 0x00000059.
- Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.
- Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.
- Count (4 bytes):** A 32-bit unsigned integer that specifies the number of points.
- aPoints (variable):** A **Count** length array of WMF **Points** objects, specified in [\[MS-WMF\]](#) section 2.2.1.12, that specifies the array of points.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.28 EMR\_POLYPOLYGON Record

The Enhanced Metafile Format (EMF) EMR\_POLYPOLYGON record specifies a series of closed polygons.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
NumberOfPolygons																															
Count																															
PolygonPointCount (variable)																															
...																															
aPoints (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYPOLYGON**. This MUST be 0x00000008.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format (WMF) **RectL** object ([\[MS-WMF\]](#) section 2.2.1.14) that specifies the bounding rectangle in device units.

**NumberOfPolygons (4 bytes):** A 32-bit unsigned integer that specifies the number of polygons.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the total number of points in all polygons.

Line width	Device supports wideline	Maximum points allowed
1	n/a	16K
> 1	yes	16K
> 1	no	1360

Any extra points MUST be ignored. To draw a line with more points, the data SHOULD be divided into groups that have less than the maximum number of points, and a EMR\_POLYPOLYGON operation SHOULD be performed for each group of points.

**PolygonPointCount (variable):** A **NumberOfPolygons** length array of 32-bit unsigned integers that specify the point counts for each polygon. Each count MUST be greater than or equal to 0x00000002.

**aPoints (variable):** A **Count** length array of WMF **PointL** objects ([\[MS-WMF\]](#) section 2.2.1.11) that specify the point data in logical units.

Each polygon SHOULD outlined using the current pen and filled using the current brush and polygon fill mode. The polygons defined by this record can overlap.

See section [2.3.5](#) for the specification of other drawing record types.

### 2.3.5.29 EMR\_POLYPOLYGON16 Record

The Enhanced Metafile Format (EMF) EMR\_POLYPOLYGON16 record specifies a series of closed polygons. Each polygon is outlined using the current pen, and filled using the current brush and polygon fill mode. The polygons drawn by this record can overlap.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
NumberOfPolygons																															
Count																															
PolygonPointCount (variable)																															
...																															
aPoints (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYPOLYGON16**. This MUST be 0x0000005B.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.

**NumberOfPolygons (4 bytes):** A 32-bit unsigned integer that specifies the number of polygons.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the total number of points in all polygons.

**PolygonPointCount (variable):** A **NumberOfPolygons** length array of 32-bit unsigned integers that specifies the point counts for each polygon.

**aPoints (variable):** A **Count** length array of WMF **Points** objects, specified in [\[MS-WMF\]](#) section 2.2.1.12, that specifies the array of points.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.30 EMR\_POLYPOLYLINE Record**

The Enhanced Metafile Format (EMF) EMR\_POLYPOLYLINE record specifies multiple series of connected line segments.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
NumberOfPolylines																															
Count																															
aPolylinePointCount (variable)																															
...																															
aPoints (variable)																															
...																															

- Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYPOLYLINE**. This MUST be 0x00000007.
- Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.
- Bounds (16 bytes):** A 128-bit Windows Metafile Format (WMF) **RectL** object ([\[MS-WMF\]](#) section 2.2.1.14) that specifies the bounding rectangle in device units.
- NumberOfPolylines (4 bytes):** A 32-bit unsigned integer that specifies the number of polylines, which is also the number of entries in the **aPolylinePointCount** array.
- Count (4 bytes):** A 32-bit unsigned integer that specifies the total number of points in all polylines.



Line width	Device supports wideline	Maximum points allowed
1	n/a	16K
> 1	yes	16K
> 1	no	1360

Any extra points MUST be ignored.

**aPolylinePointCount (variable):** A **NumberOfPolylines** length array of 32-bit unsigned integers that specify the point counts for each polyline. Each count MUST be greater than or equal to 0x00000002.

**aPoints (variable):** A **Count** length array of WMF **PointL** objects ([\[MS-WMF\]](#) section 2.2.1.11) specify the point data in logical units.

The line segments SHOULD be drawn using the current pen. The figures formed by the segments SHOULD NOT filled. The current position SHOULD neither used nor updated by this record.

See section [2.3.5](#) for the specification of other drawing record types.

### 2.3.5.31 EMR\_POLYPOLYLINE16 Record

The Enhanced Metafile Format EMR\_POLYPOLYLINE16 record specifies multiple series of connected line segments.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
NumberOfPolylines																															
Count																															
PolylinePointCount (variable)																															
...																															
aPoints (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYPOLYLINE16**. This MUST be 0x0000005A.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.

**NumberOfPolylines (4 bytes):** A 32-bit unsigned integer that specifies the number of polylines.

**Count (4 bytes):** A 32-bit unsigned integer that specifies the total number of points in all polylines.

**PolylinePointCount (variable):** A **NumberOfPolylines** length array of 32-bit unsigned integers that specifies the point counts for each polyline.

**aPoints (variable):** A **Count** length array of WMF **Points** objects, specified in [\[MS-WMF\]](#) section 2.2.1.12, that specifies the array of points.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.32 EMR\_POLYTEXTOUTA Record

The Enhanced Metafile Format EMR\_POLYTEXTOUTA record specifies several ASCII strings using the font and text colors currently selected in the current playback device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
iGraphicsMode																															
exScale																															
eyScale																															
cStrings																															
aemrtext (variable)																															
...																															

- Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYTEXTOUTA**. This MUST be 0x00000060.
- Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.
- Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.
- iGraphicsMode (4 bytes):** A 32-bit unsigned integer that specifies the current graphics mode. Graphics modes are specified in section [2.1.16](#).
- exScale (4 bytes):** A 32-bit float that specifies the X scale from page units to .01mm units if graphics mode is **GM\_COMPATIBLE**.

- eyScale (4 bytes):** A 32-bit float that specifies the X scale from page units to .01mm units if graphics mode is **GM\_COMPATIBLE**.
- cStrings (4 bytes):** A 32-bit unsigned integer that specifies the number of **EmrText** objects.
- aemrtext (variable):** An array of cStrings number of [EmrText \(section 2.2.4\)](#) objects that specify the properties of the strings to be printed, and where to find the output strings and spacing values.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.33 EMR\_POLYTEXTOUTW Record**

The Enhanced Metafile Format EMR\_POLYTEXTOUTW record specifies several UNICODE, specified in [UNICODE](#), strings using the font and text colors currently selected in the playback device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
iGraphicsMode																															
exScale																															
eyScale																															
cStrings																															
aemrtext (variable)																															
...																															

- Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_POLYTEXTOUTW**. This MUST be 0x00000061.
- Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.

**iGraphicsMode (4 bytes):** A 32-bit unsigned integer that specifies the current graphics mode. Graphics modes are specified in section [2.1.16](#).

**exScale (4 bytes):** A 32-bit float that specifies the X scale from page units to .01mm units if graphics mode is **GM\_COMPATIBLE**.

**eyScale (4 bytes):** A 32-bit float that specifies the Y scale from page units to .01mm units if graphics mode is **GM\_COMPATIBLE**.

**cStrings (4 bytes):** A 32-bit unsigned integer that specifies the number of **EmrText** objects.

**aemrtext (variable):** An array of cStrings number of [EmrText \(section 2.2.4\)](#) objects that specify the properties of the strings to be printed, and where to find the output strings and spacing values.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.34 EMR\_RECTANGLE Record**

The Enhanced Metafile Format EMR\_RECTANGLE record draws a rectangle. The rectangle is outlined by using the current pen and filled by using the current brush.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Box																															
...																															
...																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_RECTANGLE**. This MUST be 0x0000002B.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Box (16 bytes):** A 128-bit [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the inclusive-inclusive rectangle to draw.

The current position is neither used nor updated by **Rectangle**.

If a **PS\_NULL** pen is used, the dimensions of the rectangle are 1 pixel less in height and 1 pixel less in width.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.35 EMR\_ROUNDRECT Record**

The Enhanced Metafile Format EMR\_ROUNDRECT record specifies a rectangle with rounded corners. The rectangle is outlined by using the current pen and filled by using the current brush.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Box																															
...																															
...																															
...																															
Corner																															
...																															

- Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_ROUNDRECT**. This MUST be 0x0000002C.
- Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.
- Box (16 bytes):** A 128-bit [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the inclusive-inclusive bounding rectangle in logical coordinates.
- Corner (8 bytes):** A 64-bit WMF **SizeL** object, specified in [\[MS-WMF\]](#) section 2.2.1.16, which specifies the **width** and **height**, in logical coordinates, of the ellipse used to draw the rounded corners.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.36 EMR\_SETPIXELV Record**

The Enhanced Metafile Format EMR\_SETPIXELV record defines the color of the pixel at the specified logical coordinates.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Pixel																															
...																															
Color																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETPIXELV**. This MUST be 0x0000000F.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Pixel (8 bytes):** A 64-bit Windows Metafile Format **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies the logical coordinates for the pixel.

**Color (4 bytes):** A 32-bit WMF **ColorRef** object, specified in [\[MS-WMF\]](#) section 2.2.1.7, that specifies the pixel color.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.37 EMR\_SMALLTEXTOUT Record

The Enhanced Metafile Format (EMF) EMR\_SMALLTEXTOUT record outputs a string.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
x																															
y																															
cChars																															
fuOptions																															
iGraphicsMode																															
exScale																															
eyScale																															
Bounds (optional)																															
...																															
...																															
...																															
TextString (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SMALLTEXTOUT**. This MUST be 0x0000006C.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**x (4 bytes):** a 32-bit signed integer specifying the x-coordinate of where to place the string.

**y (4 bytes):** a 32-bit signed integer specifying the y-coordinate of where to place the string.

**cChars (4 bytes):** A 32-bit unsigned integer specifying the number of 16-bit characters in the string. The string is NOT null-terminated.



**fuOptions (4 bytes):** A 32-bit unsigned integer specifying the text output options to use. These options are specified by one or a combination of values from the [ExtTextOutOptions enumeration \(section 2.1.11\)](#).

**iGraphicsMode (4 bytes):** A 32-bit unsigned integer specifying the graphics mode, from the [GraphicsMode enumeration \(section 2.1.16\)](#).

**exScale (4 bytes):** A 32-bit floating-point value that specifies how much to scale the text in the x-direction.

**eyScale (4 bytes):** A 32-bit floating-point value that specifies how much to scale the text in the y-direction.

**Bounds (16 bytes):** An optional, 128-bit WMF **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.

**TextString (variable):** A variable-length string that contains the text string to draw, in either 8 or 16-bit character codes, according to the value of the **fuOptions** field.

If **ETO\_SMALL\_CHARS** is set in the **fuOptions** field, **TextString** contains 8-bit codes for characters, derived from the low bytes of 16-bit Unicode UTF16-LE character codes, in which the high byte is assumed to be 0.

If **ETO\_NO\_RECT** is set in the **fuOptions** field, the **Bounds** field is not included in the record.

See section [2.3.5](#) for the specification of other Drawing record types.

**2.3.5.38 EMR\_STROKEANDFILLPATH Record**

The Enhanced Metafile Format EMR\_STROKEANDFILLPATH record closes any open figures in a path, strokes the outline of the path by using the current pen, and fills its interior by using the current brush.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_STROKEANDFILLPATH**. This MUST be 0x0000003F.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.5.39 EMR\_STROKEPATH Record

The Enhanced Metafile Format EMR\_STROKEPATH record renders the specified path by using the current pen.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_STROKEPATH**. This MUST be 0x00000040.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.6 Escape Record Types

The Enhanced Metafile Format (EMF) escape record types execute printer driver functions.

The following are EMF escape record types:

Name	Section	Description
EMR_DRAWESCAPE	<a href="#">2.3.6.1</a>	Passes arbitrary information to the printer driver. The intent is that the information will result in drawing being done.
EMR_EXTESCAPE	<a href="#">2.3.6.2</a>	Passes arbitrary information to the printer driver. The intent is that the information will not result in drawing being done.
EMR_NAMEDESCAPE	<a href="#">2.3.6.3</a>	Passes arbitrary information to the given named printer driver.

The generic structure of EMF escape records is specified as follows:

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
iEscape																															
Data (variable)																															
...																															

**Type (4 bytes):** 32-bit unsigned integer that defines the type of the record. The escape record types are listed below. See the table above for descriptions of these records.

Name	Value
EMR_DRAWESCAPE	0x00000069
EMR_EXTESCAPE	0x0000006A
EMR_NAMEDESCAPE	0x0000006E

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**iEscape (4 bytes):** A 32-bit unsigned integer that specifies the printer driver escape to execute. This MUST be one of the values in the Windows Metafile Format (WMF) [MetafileEscapes](#) enumeration ([\[MS-WMF\]](#) section 2.1.19).

**Data (variable):** An array of bytes that contains data for the specific type of escape record.

**2.3.6.1 EMR\_DRAWESCAPE Record**

The Enhanced Metafile Format EMR\_DRAWESCAPE record passes arbitrary information to a printer driver. The intent is that the information will result in drawing being done.

**Note** Fields that are not described below are specified in [Escape Record Types \(section 2.3.6\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
iEscape																															
cjIn																															
Data (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type from the [EmrComment enumeration \(section 2.1.10\)](#). It MUST be **EMR\_DRAWESCAPE**, which is 0x00000069.

**cjIn (4 bytes):** A 32-bit unsigned integer specifying the number of bytes to pass to the printer driver.

**Data (variable):** The data to pass to the printer driver. There MUST be **cjIn** bytes available.

See section [2.3.6](#) for the specification of other escape record types.

### 2.3.6.2 EMR\_EXTESCAPE Record

The Enhanced Metafile Format EMR\_EXTESCAPE record passes arbitrary information to a printer driver. The intent is that the information will not result in drawing being done.

**Note** Fields that are not described below are specified in [Escape Record Types \(section 2.3.6\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
iEscape																															
cjIn																															
Data (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type from the [EmrComment enumeration \(section 2.1.10\)](#). It MUST be **EMR\_EXTESCAPE**, which is 0x0000006A.

**cjIn (4 bytes):** A 32-bit unsigned integer specifying the number of bytes to pass to the printer driver.

**Data (variable):** The data to pass to the printer driver. There MUST be **cjIn** bytes available.

See section [2.3.6](#) for the specification of other escape record types.

### 2.3.6.3 EMR\_NAMEDESCAPE Record

The Enhanced Metafile Format MR\_NAMEDESCAPE record passes arbitrary information to a specified printer driver.

**Note** Fields that are not described below are specified in [Escape Record Types \(section 2.3.6\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
iEscape																															
cjDriver																															
cjIn																															
DriverName (variable)																															
...																															
Data (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type from the [EmrComment enumeration \(section 2.1.10\)](#). It MUST be **EMR\_NAMEDESCAPE**, which is 0x0000006E.

**cjDriver (4 bytes):** A 32-bit unsigned integer that specifies the number of bytes in the **DriverName** field. This value MUST be an even number.

**cjIn (4 bytes):** A 32-bit unsigned integer specifying the number of bytes to pass to the printer driver.

**DriverName (variable):** A string of 16-bit Unicode characters that specify the name of the printer driver that will receive data. This value MUST be **cjDriver** bytes long, and it MUST be terminated with a null character.

**Data (variable):** The data to pass to the printer driver. There MUST be **cjIn** bytes available.

See section [2.3.6](#) for the specification of other escape record types.

### 2.3.7 Object Creation Record Types

The Enhanced Metafile Format (EMF) object creation record types create graphics objects.

The following are EMF object creation record types:

Name	Section	Description
EMR_CREATEBRUSHINDIRECT	<a href="#">2.3.7.1</a>	Specifies a logical brush that has the specified style, color, and pattern.

Name	Section	Description
EMR_CREATECOLORSPACE	<a href="#">2.3.7.2</a>	Specifies a logical mapping of color components onto a geometric coordinate system in three dimensions.
EMR_CREATECOLORSPACEW	<a href="#">2.3.7.3</a>	Specifies a logical color space.
EMR_CREATEDIBPATTERNBRUSHPT	<a href="#">2.3.7.4</a>	Specifies a logical brush that has a pattern specified by a DIB.
EMR_CREATEMONOBRUSH	<a href="#">2.3.7.5</a>	Specifies a logical brush with the specified bitmap pattern.
EMR_CREATEPALETTE	<a href="#">2.3.7.6</a>	Creates an empty <a href="#">LogPalette (section 2.2.15)</a> object.
EMR_CREATEPEN	<a href="#">2.3.7.7</a>	Specifies a logical pen that has the specified style, width, and color.
EMR_EXTCREATEFONTINDIRECTW	<a href="#">2.3.7.8</a>	Specifies a logical font that has the specified characteristics.
EMR_EXTCREATEPEN	<a href="#">2.3.7.9</a>	Specifies a logical cosmetic or geometric pen that has the specified style, width, and brush attributes.

The generic structure of EMF object creation records is specified as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
dParm (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that defines the type of record. The object creation record types are listed below. See the table above for descriptions of these record types.

Name	Value
EMR_CREATEMONOBRUSH	0x0000005D
EMR_CREATEDIBPATTERNBRUSHPT	0x0000005E
EMR_EXTCREATEPEN	0x0000005F
EMR_CREATECOLORSPACEW	0x0000007A
EMR_CREATEPEN	0x00000026
EMR_CREATEBRUSHINDIRECT	0x00000027

Name	Value
EMR_CREATEPALETTE	0x00000031
EMR_EXTCREATEFONTINDIRECTW	0x00000052
EMR_CREATECOLORSPACE	0x00000063

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size, in bytes, of the **dParm** field in the record.

**dParm (variable):** An array of 32-bit integers that contains parameters for the specific type of object creation record.

### 2.3.7.1 EMR\_CREATEBRUSHINDIRECT Record

The Enhanced Metafile Format EMR\_CREATEBRUSHINDIRECT record specifies a logical brush that has the specified style, color, and pattern. The **BrushStyle** MUST be **BS\_SOLID**, **BS\_HATCHED**, or **BS\_NULL**.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihBrush																															
LogBrush																															
...																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_CREATEBRUSHINDIRECT**. This MUST be 0x00000027.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihBrush (4 bytes):** A 32-bit unsigned integer that specifies the brush [EMF Object Table](#) index.

**LogBrush (12 bytes):** A 96-bit [LogBrushEx \(section 2.2.10\)](#) object that specifies brush data.

See section [2.3.7](#) for the specification of other Object Creation record types.

### 2.3.7.2 EMR\_CREATECOLORSPACE Record

The Enhanced Metafile Format EMR\_CREATECOLORSPACE record specifies a logical mapping of color components onto a geometric coordinate system in three dimensions. [<41>](#)



0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Type																															
Size																															
ihCS																															
lcs (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_CREATECOLORSPACE**. This MUST be 0x00000063.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihCS (4 bytes):** A 32-bit unsigned integer that specifies the **ColorSpace** [EMF Object Table](#) index.

**lcs (variable):** A **LogColorSpace** object, specified in [\[MS-WMF\]](#) section 2.2.2.5, which specifies the ASCII version logical color space record.

See section [2.3.7](#) for the specification of other Object Creation record types.

### 2.3.7.3 EMR\_CREATECOLORSPACEW Record

The Enhanced Metafile Format EMR\_CREATECOLORSPACEW record specifies a logical color space. [<42>](#)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihCS																															
lcs (variable)																															
...																															
dwFlags																															
cbData																															
Data (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_CREATECOLORSPACEW**. This MUST be 0x0000007A.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihCS (4 bytes):** A 32-bit unsigned integer that specifies the **ColorSpace** [EMF Object Table](#) index.

**lcs (variable):** A variable-length **LogColorSpaceW** object, specified in [\[MS-WMF\]](#) section 2.2.2.6, that specifies the Unicode, as specified in [\[UNICODE\]](#), version of a logical color space record.

**dwFlags (4 bytes):** A 32-bit unsigned integer that specifies the values.

**cbData (4 bytes):** A 32-bit unsigned integer that specifies the size of the raw source profile data, if attached.

**Data (variable):** A variable-length array that specifies the source profile data.

See section [2.3.7](#) for the specification of other Object Creation record types.

#### 2.3.7.4 EMR\_CREATEDIBPATTERNBRUSHPT Record

The Enhanced Metafile Format EMR\_CREATEDIBPATTERNBRUSHPT record specifies a logical brush that has a pattern specified by a DIB.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihBrush																															
Usage																															
offBmi																															
cbBmi																															
offBits																															
cbBits																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as EMR\_CREATEDIBPATTERNBRUSHPT. This MUST be 0x0000005E.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihBrush (4 bytes):** A 32-bit unsigned integer that specifies the brush [EMF Object Table](#) index.

**Usage (4 bytes):** A 32-bit unsigned integer that specifies the value of the **Colors** field of the [Windows Metafile Format DeviceIndependentBitmap](#) object, specified in [\[MS-WMF\]](#) section 2.2.2.3. The value MUST be in the [DIBColors \(section 2.1.9\)](#) enumeration.

**offBmi (4 bytes):** A 32-bit unsigned integer that specifies the offset to the WMF **DeviceIndependentBitmap** object.

**cbBmi (4 bytes):** A 32-bit unsigned integer that specifies the size of the WMF **DeviceIndependentBitmap** object.

**offBits (4 bytes):** A 32-bit unsigned integer that specifies the offset to the bitmap bits.

**cbBits (4 bytes):** A 32-bit unsigned integer that specifies the size of the bitmap bits.

See section [2.3.7](#) for the specification of other Object Creation record types.

### 2.3.7.5 EMR\_CREATEMONOBRUSH Record

The Enhanced Metafile Format EMR\_CREATEMONOBRUSH record specifies a logical brush with the specified bitmap pattern. The bitmap can be a DIB section bitmap or it can be a device-dependent bitmap.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihBrush																															
Usage																															
offBmi																															
cbBmi																															
offBits																															
cbBits																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_CREATEMONOBRUSH**. This MUST be 0x0000005D.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihBrush (4 bytes):** A 32-bit unsigned integer that specifies the brush [EMF Object Table](#) index.

**Usage (4 bytes):** A 32-bit unsigned integer that specifies the value of the **Colors** field of the [Windows Metafile Format DeviceIndependentBitmap](#) object, specified in [\[MS-WMF\]](#) section 2.2.2.3. The value MUST be in the [DIBColors \(section 2.1.9\)](#) enumeration.

**offBmi (4 bytes):** A 32-bit unsigned integer that specifies the offset to the WMF **DeviceIndependentBitmap** object.

**cbBmi (4 bytes):** A 32-bit unsigned integer that specifies the size of the WMF **DeviceIndependentBitmap** object.

**offBits (4 bytes):** A 32-bit unsigned integer that specifies the offset to the bitmap bits.

**cbBits (4 bytes):** A 32-bit unsigned integer that specifies the size of the bitmap bits.

See section [2.3.7](#) for the specification of other Object Creation record types.

#### 2.3.7.6 EMR\_CREATEPALETTE Record

The Enhanced Metafile Format EMR\_CREATEPALETTE record creates an empty [LogPalette \(section 2.2.15\)](#) object.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihPal																															
LogPalette (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_CREATEPALETTE**. This MUST be 0x00000031.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihPal (4 bytes):** A 32-bit unsigned integer that specifies the palette [EMF Object Table](#) index. This index MUST be saved so that this LogPalette object can be reused or modified.

**LogPalette (variable):** A variable-length LogPalette object. The **Version** field of the LogPalette object MUST be set to 0x0300.

If the LogPalette object is empty, with its length field set to zero, processing of this record MUST fail.

See section [2.3.7](#) for the specification of other Object Creation record types.

### 2.3.7.7 EMR\_CREATEPEN Record

The Enhanced Metafile Format EMR\_CREATEPEN record specifies a logical pen that has the specified style, width, and color. The pen can subsequently be selected into a playback device context and used to draw lines and curves.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihPen																															
LogPen																															
...																															
...																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_CREATEPEN**. This MUST be 0x00000026.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihPen (4 bytes):** A 32-bit unsigned integer that specifies the pen [EMF Object Table](#) index.

**LogPen (16 bytes):** A 128-bit [LogPen \(section 2.2.17\)](#) object that specifies the pen data.

See section [2.3.7](#) for the specification of other Object Creation record types.

### 2.3.7.8 EMR\_EXTCREATEFONTINDIRECTW Record

The Enhanced Metafile Format (EMF) EMR\_EXTCREATEFONTINDIRECTW record creates a logical font object from specified logical font data. The font object can subsequently be selected as the current font in the metafile playback device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihFonts																															
elw (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_EXTCREATEFONTINDIRECTW**. This MUST be 0x00000052.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihFonts (4 bytes):** A 32-bit unsigned integer that specifies the font index in the [EMF Object Table](#).

**elw (variable):** A variable-length field that contains a [LogFont](#) or [LogFontExDv](#) object, which specifies the logical font. The structures of these objects are specified in sections [2.2.11](#) and [2.2.13](#), respectively.

In an **EMR\_EXTCREATEFONTINDIRECTW** record, the type of logical font object in the **elw** field is determined by the following algorithm (all size and length values are in bytes):

- First, note that the size in bytes of the static part of this record—that is, the sum of the sizes of its **Type**, **Size**, and **ihFonts** fields—is 12.
- Next, note that because the size in bytes of the entire record is present in its **Size** field, the size in bytes of the variable-length **elw** field can be computed as:

Size - 12

- If the size of the **elw** field is equal to or less than the size of a [LogFontPanoose](#) structure, **elw** MUST be treated as a fixed-length LogFont object. Bytes beyond the extent of the LogFont object, up to the end of the **elw** field, MUST be ignored.
- If the size of the **elw** field is greater than the size of a LogFontPanoose structure, then **elw** MUST be treated as a variable-length LogFontExDv object.

The size of the LogFontPanoose structure is 0x0140 (320 decimal). It is determined by adding up the sizes of its fields, as follows:

- LogFont:** The size of the LogFont structure is 0x005C (92 decimal). It is determined by adding up the sizes of its fields, as follows:
  - Fields from **Height** through **PitchAndFamily**: 0x001C (28 decimal).

- **Facename:** The length is 32 16-bit characters: 0x0040 (64 decimal).
- **Fullname:** The length is 64 16-bit characters: 0x0080 (128 decimal).
- **Style:** The length is 32 16-bit characters: 0x0040 (64 decimal).
- Fields from **Version** through **Culture:** 0x0018 (24 decimal).
- **Panose:** The exact length of this field is 0x000A, but it MUST be padded by two additional bytes for 32-bit alignment, so for the purposes of this computation the length is 0x000C (12 decimal)

See section [2.2.14](#) for further information concerning the LogFontPanose structure.

See section [2.3.7](#) for the specification of other Object Creation record types.

### 2.3.7.9 EMR\_EXTCREATEPEN Record

The Enhanced Metafile Format (EMF) EMR\_EXTCREATEPEN record specifies a logical cosmetic or geometric pen that has the specified style, width, and brush attributes.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihPen																															
offBmi																															
cbBmi																															
offBits																															
cbBits																															
elp (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_EXTCREATEPEN**. This MUST be 0x0000005F.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihPen (4 bytes):** A 32-bit unsigned integer that specifies the pen [EMF Object Table](#) index.



**offBmi (4 bytes):** A 32-bit unsigned integer that specifies the offset to the [Windows Metafile Format \(WMF\) Device](#) object, specified in [\[MS-WMF\]](#) section 2.2.2.3, if the record contains a bitmap.

**cbBmi (4 bytes):** A 32-bit unsigned integer that specifies the size of the WMF **DeviceIndependentBitmap** object, if the record contains a bitmap.

**offBits (4 bytes):** A 32-bit unsigned integer that specifies the offset to the bitmap bits, if the record contains a bitmap.

**cbBits (4 bytes):** A 32-bit unsigned integer that specifies the size of the bitmap bits, if the record contains a bitmap.

**elp (variable):** A [LogPenEx](#) object that specifies the extended pen with the style array.

See section [2.3.7](#) for the specification of other Object Creation record types.

### 2.3.8 Object Manipulation Record Types

The Enhanced Metafile Format (EMF) object manipulation record types manage and modify graphics objects.

The following are EMF object manipulation record types:

Name	Section	Description
EMR_COLORCORRECTPALETTE	<a href="#">2.3.8.1</a>	Specifies how to correct the entries of a <a href="#">LogPalette</a> ( <a href="#">section 2.2.15</a> ) object, using the WCS 1.0 members in the playback device context.
EMR_DELETECOLORSPACE	<a href="#">2.3.8.2</a>	Specifies how to delete a logical color space from the <a href="#">EMF Object Table</a> .
EMR_DELETEOBJECT	<a href="#">2.3.8.3</a>	Specifies the index of the object to be deleted from the EMF Object Table.
EMR_RESIZEPALETTE	<a href="#">2.3.8.4</a>	Increases or decreases the size of an existing LogPalette object.
EMR_SELECTOBJECT	<a href="#">2.3.8.5</a>	Specifies an existing object based on its index in the EMF Object Table.
EMR_SELECTPALETTE	<a href="#">2.3.8.6</a>	Selects the specified LogPalette object into the playback device context.
EMR_SETCOLORSPACE	<a href="#">2.3.8.7</a>	Specifies a logical color space, based on its index in the EMF Object Table.
EMR_SETPALETTEENTRIES	<a href="#">2.3.8.8</a>	Defines Red Green Blue (RGB) color values in a range of entries for an existing LogPalette object.

The generic structure of EMF object manipulation records is specified as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
dParm (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that defines the type of record. The object manipulation record types are listed below. See the table above for descriptions of these record types.

Name	Value
EMR_SELECTOBJECT	0x00000025
EMR_DELETEOBJECT	0x00000028
EMR_SELECTPALETTE	0x00000030
EMR_SETPALETTEENTRIES	0x00000032
EMR_RESIZEPALETTE	0x00000033
EMRSetColorSpace	0x00000064
EMR_DELETECOLORSPACE	0x00000065
EMR_COLORCORRECTPALETTE	0x0000006F

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size, in bytes, of the **dParm** field in the record.

**dParm (variable):** An array of 32-bit integers that contains parameters for the specific type of object manipulation record.

### 2.3.8.1 EMR\_COLORCORRECTPALETTE Record

The Enhanced Metafile Format EMR\_COLORCORRECTPALETTE record specifies how to correct the entries of a [LogPalette \(section 2.2.15\)](#) object, using the WCS 1.0 members in the playback device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihPalette																															
nFirstEntry																															
nPalEntries																															
nReserved																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_COLORCORRECTPALETTE**. This MUST be 0x0000006F.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihPalette (4 bytes):** A 32-bit unsigned integer that specifies the [EMF Object Table](#) index for the LogPalette object.

**nFirstEntry (4 bytes):** A 32-bit unsigned integer that specifies the index of the first entry to correct.

**nPalEntries (4 bytes):** A 32-bit unsigned integer that specifies the number of palette entries to correct.

**nReserved (4 bytes):** A 32-bit unsigned integer that is undefined and unused.

See section [2.3.8](#) for the specification of other Object Manipulation record types.

### 2.3.8.2 EMR\_DELETECOLORSPACE Record

The Enhanced Metafile Format EMR\_DELETECOLORSPACE record specifies how to delete a logical color space from the [EMF Object Table](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihCS																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_DELETECOLORSPACE**. This MUST be 0x00000065.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihCS (4 bytes):** A 32-bit unsigned integer that specifies the **ColorSpace** EMF Object Table index.

See section [2.3.8](#) for the specification of other Object Manipulation record types.

### 2.3.8.3 EMR\_DELETEOBJECT Record

The Enhanced Metafile Format (EMF) EMR\_DELETEOBJECT record deletes a graphics object, which is specified by its index in the [EMF Object Table \(section 3.1.1.1\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihObject																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_DELETEOBJECT**. This MUST be 0x00000028.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihObject (4 bytes):** A 32-bit unsigned integer that specifies the index of a graphics object in the EMF Object Table.

This value MUST NOT be 0, which is a reserved index that refers to the EMF metafile itself; and it MUST NOT be the index of a stock object, which cannot be deleted. Stock object indexes are specified in the [StockObject \(section 2.1.31\)](#) enumeration.

The object specified by this record MUST be deleted from the EMF Object Table. If the deleted object is currently selected in the playback device context, the default object for that graphics property MUST be restored.

See section [2.3.8](#) for the specification of other Object Manipulation record types.

#### 2.3.8.4 EMR\_RESIZEPALETTE Record

The Enhanced Metafile Format [EMR\\_RESIZEPALETTE](#) record increases or decreases the size of an existing [LogPalette \(section 2.2.15\)](#) object.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihPal																															
NumberOfEntries																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_RESIZEPALETTE**. This MUST be 0x00000033.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihPal (4 bytes):** A 32-bit unsigned integer that specifies the palette [EMF Object Table](#) index.

**NumberOfEntries (4 bytes):** A 32-bit unsigned integer that specifies the number of entries in the palette after resizing. The value MUST be less than or equal to 0x00000400 and greater than 0x00000000.

The new size of the LogPalette object MUST be reflected in the **NumberOfEntries** field in that structure.

See section [2.3.8](#) for the specification of other Object Manipulation record types.

#### 2.3.8.5 EMR\_SELECTOBJECT Record

The Enhanced Metafile Format (EMF) EMR\_SELECTOBJECT record adds a graphics object to the current metafile playback device context. The object is specified either by its index in the [EMF Object Table \(section 3.1.1.1\)](#) or by its value from the [StockObject enumeration \(section 2.1.31\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihObject																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SELECTOBJECT**. This MUST be 0x00000025.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihObject (4 bytes):** A 32-bit unsigned integer that specifies either the index of a graphics object in the EMF Object Table, or the index of a stock object from the **StockObject enumeration**.

This value MUST NOT be 0, which is a reserved index that refers to the EMFmetafile itself.

The object specified by this record MUST be used in drawing operations that require such an object, until a different object of the same type is specified by another EMR\_SELECTOBJECT record, or the object is removed by an [EMR\\_DELETEOBJECT](#) record.

See section [2.3.8](#) for the specification of other Object Manipulation record types.

### 2.3.8.6 EMR\_SELECTPALETTE Record

The Enhanced Metafile Format (EMF) EMR\_SELECTPALETTE record selects the specified [LogPalette object \(section 2.2.15\)](#) into a playback device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihPal																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SELECTPALETTE**. This MUST be 0x00000030.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihPal (4 bytes):** A 32-bit unsigned integer that specifies either the index of a palette object in the [EMF Object Table](#), or the value **DEFAULT\_PALETTE**, which is the index of a stock object from the [StockObject enumeration \(section 2.1.31\)](#).

This value MUST NOT be 0, which is a reserved index that refers to the EMFmetafile itself; and it MUST NOT be the index of any other stock object.

See section [2.3.8](#) for the specification of other Object Manipulation record types.

### 2.3.8.7 EMR\_SETCOLORSPACE Record

The Enhanced Metafile Format EMR\_SETCOLORSPACE record specifies a logical color space, based on its index in the [EMF Object Table](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihCS																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETCOLORSPACE**. This MUST be 0x00000064.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihCS (4 bytes):** A 32-bit unsigned integer that specifies the EMF Object Table index of a [Windows Metafile Format LogColorSpace](#) or [LogColorSpaceW](#) object. These objects are specified in [MS-WMF] sections 2.2.2.5 and 2.2.2.6, respectively.

The object specified by this record MUST be used in drawing operations that require a logical color space object, until a different color space object is specified by another EMR\_SETCOLORSPACE record, or the object is removed by a [EMR\\_DELETEOBJECT](#) record.

See section [2.3.8](#) for the specification of other Object Manipulation record types.

### 2.3.8.8 EMR\_SETPALETTEENTRIES Record

The Enhanced Metafile Format EMR\_SETPALETTEENTRIES record defines RGB (red, green, blue) color values in a range of entries for an existing [LogPalette \(section 2.2.15\)](#) object.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ihPal																															
Start																															
NumberOfEntries																															
aPalEntries																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETPALETTEENTRIES**. This MUST be 0x00000032.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ihPal (4 bytes):** A 32-bit unsigned integer that specifies the palette [EMF Object Table](#) index.

**Start (4 bytes):** A 32-bit unsigned integer that specifies the index of the first entry to set.

**NumberOfEntries (4 bytes):** A 32-bit unsigned integer that specifies the number of entries.

**aPalEntries (4 bytes):** An array of [LogPaletteEntry \(section 2.2.16\)](#) objects, of **NumberOfEntries** length that specifies the palette entry data. The **Values** members do not contain any values.

See section [2.3.8](#) for the specification of other Object Manipulation record types.

### 2.3.9 OpenGL Record Types

The Enhanced Metafile Format (EMF) OpenGL record types specify metafile records generated by OpenGL.

The following are EMF OpenGL record types:

Name	Section	Description
EMR_GLSBOUNDEDRECORD	<a href="#">2.3.9.1</a>	Specifies an enhanced metafile record generated by OpenGL functions within a bounding rectangle.
EMR_GLSRECORD	<a href="#">2.3.9.2</a>	Specifies an enhanced metafile record generated by OpenGL functions.

The generic structure of EMF OpenGL records is specified as follows:



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
dParm (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that defines the type of record. The OpenGL record types are listed below. See the table above for descriptions of these record types, and [\[OPENGL\]](#) for further information about OpenGL technology.

Name	Value
EMR_GLSRECORD	0x00000066
EMR_GLSBOUNDEDRECORD	0x00000067

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size, in bytes, of the **dParm** field in the record.

**dParm (variable):** An array of 32-bit integers that contains parameters for the specific type of OpenGL record.

### 2.3.9.1 EMR\_GLSBOUNDEDRECORD Record

The Enhanced Metafile Format EMR\_GLSBOUNDEDRECORD record specifies an enhanced metafile record generated by OpenGL functions within a bounding rectangle. It contains data for OpenGL functions with information in pixel units that **MUST** be scaled when playing the metafile.

0	1	2	3	4	5	6	7	8	9	0 <sup>1</sup>	1	2	3	4	5	6	7	8	9	0	1 <sup>2</sup>	0	1	2	3	4	5	6	7	8	9	0	1 <sup>3</sup>	0	1
Type																																			
Size																																			
Bounds																																			
...																																			
...																																			
...																																			
cbData																																			
Data (variable)																																			
...																																			

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_GLSBOUNDEDRECORD**. This MUST be 0x00000067.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit Windows Metafile Format **RectL** object, specified in [\[MS-WMF\]](#) section 2.2.1.14, that specifies the bounding rectangle in device units.

**cbData (4 bytes):** A 32-bit unsigned integer that specifies the size of the raw profile data, if attached.

**Data (variable):** A byte array of **cbData** length that specifies the OpenGL function to be performed.

### 2.3.9.2 EMR\_GLSRECORD Record

The Enhanced Metafile Format EMR\_GLSRECORD record specifies an enhanced metafile record generated by OpenGL functions. It contains data for OpenGL functions that scale automatically to the OpenGL viewport.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
cbData																															
Data (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_GLSRECORD**. This MUST be 0x00000066.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**cbData (4 bytes):** A 32-bit unsigned integer that specifies the size of the raw profile data, if attached.

**Data (variable):** A byte array of **cbData** length that specifies the OpenGL function to be performed.

### 2.3.10 Path Bracket Record Types

The Enhanced Metafile Format (EMF) path bracket record types specify and manipulate paths in path brackets.

**Note** None of the EMF path bracket records specify parameters.

The following are EMF path bracket record types:

Name	Description
EMR_ABORTPATH	This record aborts a path bracket or discards the path from a closed path bracket.
EMR_BEGINPATH	<p>This record opens a path bracket in the current <b>device context</b>.</p> <p>After a path bracket is open, an application can begin processing records to define the points that lie in the path. An application MUST close an open path bracket by processing the <b>EMR_ENDPATH</b> record.</p> <p>When an application processes the <b>EMR_BEGINPATH</b> record for a device context, any previous paths MUST be discarded from that device context.</p>
EMR_CLOSEFIGURE	<p>This record closes an open figure in a path.</p> <p>Processing the <b>EMR_CLOSEFIGURE</b> record MUST close the figure by drawing a line from the current position to the first point of the figure, and then it MUST connect the lines by using the line join style. If a figure is closed by processing the <b>EMR_LINETO</b> record instead of the <b>EMR_CLOSEFIGURE</b> record, end caps are used to create the corner instead of a join. <b>EMR_LINETO</b> is specified in section <a href="#">2.3.5.13</a>.</p> <p>The <b>EMR_CLOSEFIGURE</b> record SHOULD only be used if there is an open path</p>

Name	Description
	bracket in the device context. A figure in a path is open unless it is explicitly closed by processing this record. Note: A figure can be open even if the current point and the starting point of the figure are the same. After processing the <b>EMR_CLOSEFIGURE</b> record, adding a line or curve to the path <b>MUST</b> start a new figure.
EMR_ENDPATH	This record closes a path bracket and selects the path defined by the bracket into the device context.
EMR_FLATTENPATH	This record transforms any curves in the path that is selected into the device context; each curve <b>MUST</b> be turned into a sequence of lines.
EMR_WIDENPATH	This record redefines the current path as the area that would be painted if the path were stroked using the pen currently selected into the device context.

The generic structure of EMF path bracket records is specified as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															

**Type (4 bytes):** 32-bit unsigned integer that defines the type of the record. The types of records that specify no parameters are listed below. See the table above for descriptions of these records.

Name	Value
EMR_BEGINPATH	0x0000003B
EMR_ENDPATH	0x0000003C
EMR_CLOSEFIGURE	0x0000003D
EMR_FLATTENPATH	0x00000041
EMR_WIDENPATH	0x00000042
EMR_ABORTPATH	0x00000044

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size, in bytes, of the record. For path bracket records, this value **MUST** be 0x00000008

### 2.3.11 State Record Types

The Enhanced Metafile Format (EMF) state record types specify and manage graphics properties.

The following are EMF state record types:

Name	Section	Description
EMR_COLORMATCHTOTARGETW	<a href="#">2.3.11.1</a>	Specifies how to preview colors as they would appear on the target device.
EMR_FORCEUFIMAPPING	<a href="#">2.3.11.2</a>	Forces the font mapper to match fonts based on their <b>UniversalFontId</b> in preference to their <a href="#">LogFont (section 2.2.11)</a> information.
EMR_INVERTGRN	<a href="#">2.3.11.3</a>	Inverts the colors in the specified region.
EMR_MOVETOEX	<a href="#">2.3.11.4</a>	Specifies the coordinates of a new drawing position, in logical units.
EMR_PIXELFORMAT	<a href="#">2.3.11.5</a>	Specifies the pixel format.
EMR_REALIZEPALETTE	2.3.11	This record maps palette entries from the current <a href="#">LogPalette object (section 2.2.15)</a> to the system_palette. This EMF record specifies no parameters.
EMR_RESTOREDC	<a href="#">2.3.11.6</a>	Restores the playback device context to the specified state, which was saved by a preceding <b>EMR_SAVEDC</b> record.
EMR_SAVEDC	2.3.11	Saves the current state of playback device context on a stack of states saved by preceding <b>EMR_SAVEDC</b> records, if any. The state consists of graphics properties and objects, including the currently selected bitmap, brush, palette, font, pen, and region. An <b>EMR_RESTOREDC</b> record is used to restore the state. This EMF record specifies no parameters.
EMR_SCALEVIEWPORTEXTEX	<a href="#">2.3.11.7</a>	Respecifies the viewport for the playback device context by using the ratios formed by the specified multiplicands and divisors.
EMR_SCALEWINDOWEXTEX	<a href="#">2.3.11.8</a>	Respecifies the window for the playback device context by using the ratios formed by the specified multiplicands and divisors.
EMR_SETARCDIRECTION	<a href="#">2.3.11.9</a>	Specifies the drawing direction to be used for arc and rectangle output.
EMR_SETBKCOLOR	<a href="#">2.3.11.10</a>	Specifies the background color.
EMR_SETBKMODE	<a href="#">2.3.11.11</a>	Specifies the background mode, which determines how to combine the background with foreground text, hatched brushes, and pen styles that are not solid lines.
EMR_SETBRUSHORGE	<a href="#">2.3.11.12</a>	Specifies the origin of the current brush.
EMRSetColorAdjustment	<a href="#">2.3.11.13</a>	Specifies color adjustment values to use in bitmap stretching.
EMR_SETICMMode	<a href="#">2.3.11.14</a>	Specifies Image Color Management (ICM) to be enabled, disabled, or queried on the playback device context.
EMR_SETICMProfile	<a href="#">2.3.11.15</a>	Specifies how to set a specified color profile as the output profile for the playback device context.

Name	Section	Description
EMR_SETICMPROFILEW	<a href="#">2.3.11.16</a>	Specifies how to set a specified color profile as the output profile for the playback device context.
EMR_SETLAYOUT	<a href="#">2.3.11.17</a>	Respecifies the layout of the playback device context.
EMR_SETLINKEDUFIS	<a href="#">2.3.11.18</a>	Sets the <a href="#">UniversalFontIds (section 2.2.24)</a> of the linked fonts to use during character lookup.
EMR_SETMAPMODE	<a href="#">2.3.11.19</a>	Specifies the mapping mode for the playback device context.
EMR_SETMAPPERFLAGS	<a href="#">2.3.11.20</a>	Respecifies the algorithm the font mapper uses when it maps logical fonts to physical fonts.
EMR_SETMITERLIMIT	<a href="#">2.3.11.21</a>	Specifies the limit for the length of miter joins for the playback device context.
EMR_SETPOLYFILLMODE	<a href="#">2.3.11.22</a>	Defines polygon fill mode.
EMR_SETROP2	<a href="#">2.3.11.23</a>	Defines a binary raster operation mode.
EMR_SETSTRETCHBLTMODE	<a href="#">2.3.11.24</a>	Specifies bitmap stretch mode.
EMR_SETTEXTALIGN	<a href="#">2.3.11.25</a>	Specifies text alignment.
EMR_SETTEXTCOLOR	<a href="#">2.3.11.26</a>	Defines the current text color.
EMR_SETTEXTJUSTIFICATION	<a href="#">2.3.11.27</a>	Sets the amount of extra space to add to break characters for justification purposes.
EMR_SETVIEWPORTEXT	<a href="#">2.3.11.28</a>	Defines the viewport extent.
EMR_SETVIEWPORTORIGEX	<a href="#">2.3.11.29</a>	Defines the viewport origin.
EMR_SETWINDOWEXT	<a href="#">2.3.11.30</a>	Defines the window extent.
EMR_SETWINDOWORIGEX	<a href="#">2.3.11.31</a>	Defines the window origin.

The generic structure of EMF state records is specified as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
dParm (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that defines the type of record. The state record types are listed below. See the table above for descriptions of these record types.

Name	Value
EMR_SETWINDOWORGE	0x0000000A
EMR_SETVIEWPORTE	0x0000000B
EMR_SETVIEWPORTORGE	0x0000000C
EMR_SETBRUSHORGE	0x0000000D
EMR_SETCOLORADJUSTMENT	0x00000017
EMR_MOVETOEX	0x0000001B
EMR_SCALEVIEWPORTE	0x0000001F
EMR_SETWINDOWEXTE	0x00000009
EMR_SETMAPPERFLAGS	0x00000010
EMR_SETMAPMODE	0x00000011
EMR_SETBKMODE	0x00000012
EMR_SETPOLYFILLMODE	0x00000013
EMR_SETROP2	0x00000014
EMR_SETSTRETCHBLTMODE	0x00000015
EMR_SETTEXTALIGN	0x00000016
EMR_SETTEXTCOLOR	0x00000018
EMR_SETBKCOLOR	0x00000019
EMR_SCALEWINDOWEXTE	0x00000020
EMR_SAVEDC	0x00000021
EMR_RESTOREDC	0x00000022
EMR_REALIZEPALETTE	0x00000034
EMR_SETARCDIRECTION	0x00000039
EMR_SETMITERLIMIT	0x0000003A
EMR_INVERTGRN	0x00000049
EMR_SETICMMODE	0x00000062
EMR_PIXELFORMAT	0x00000068
EMR_FORCEUFIMAPPING	0x0000006D
EMR_SETICMPROFILEA	0x00000070
EMR_SETICMPROFILEW	0x00000071

Name	Value
EMR_SETLAYOUT	0x00000073
EMR_SETLINKEDUFIS	0x00000077
EMR_SETTEXTJUSTIFICATION	0x00000078
EMR_COLORMATCHTOTARGETW	0x00000079

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size, in bytes, of the **dParm** field in the record.

**dParm (variable):** An optional array of 32-bit unsigned integers that specify parameters for the state record.

### 2.3.11.1 EMR\_COLORMATCHTOTARGETW Record

The Enhanced Metafile Format (EMF) EMR\_COLORMATCHTOTARGETW record specifies how to preview colors as they would appear on the target device.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
dwAction																															
dwValues																															
cbName																															
cbData																															
Data (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as EMR\_COLORMATCHTOTARGETW. This MUST be 0x00000079.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**dwAction (4 bytes):** A 32-bit unsigned integer that specifies a value from the [ColorSpace \(section 2.1.7\)](#) enumeration.

**dwValues (4 bytes):** A 32-bit unsigned integer that specifies a values from the [ColorMatchToTarget \(section 2.1.6\)](#) enumeration.



**cbName (4 bytes):** A 32-bit unsigned integer that specifies the size in bytes of the desired [\[UNICODE\]](#) target color profile name.

**cbData (4 bytes):** A 32-bit unsigned integer that specifies the size of the raw target color profile data, if it is attached.

**Data (variable):** A byte array of size (cbName + cbData) that specifies the [\[UNICODE\]](#) color profile name and data.

A **EMR\_COLORMATCHTOTARGETW** record can be used for color proofing the colors of one output device on another output device. If the **dwAction** value is **CS\_ENABLE**, all subsequent drawing to the playback device context MUST render colors as they would appear on the target device. If **dwAction** is set to **CS\_DISABLE**, proofing MUST be turned off. In this case, the current color transform is not deleted from the playback device context. It is just inactive.

When a **EMR\_COLORMATCHTOTARGETW** record is processed, the color transform for the target device is performed first, and then the transform to the playback device context is applied to the results of the first transform. This can be used for checking gamut mapping conditions. [<43>](#)

This process cannot be cascaded. While color mapping to the target is enabled by a **dwAction** value of **CS\_ENABLE**, changes to the color space or gamut mapping method are ignored. However, those changes MUST take effect when color mapping to the target is disabled.

The **dwAction** field SHOULD NOT be set to **CS\_DELETE\_TRANSFORM** unless color management has already been enabled.

See section [2.3.11](#) for the specification of other State record types.

**2.3.11.2 EMR\_FORCEUFIMAPPING Record**

The Enhanced Metafile Format **EMR\_FORCEUFIMAPPING** record forces the font mapper to match fonts based on their **UniversalFontId** in preference to their [LogFont \(section 2.2.11\)](#) information.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ufi																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_FORCEUFIMAPPING**. This MUST be 0x0000006D.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ufi (8 bytes):** The font id to use, specified as a [UniversalFontId \(section 2.2.24\)](#).

See section [2.3.5](#) for the specification of other Drawing record types.

### 2.3.11.3 EMR\_INVERTGRN Record

The Enhanced Metafile Format EMR\_INVERTGRN record inverts the colors in the specified region.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Bounds																															
...																															
...																															
...																															
RgnDataSize																															
RgnData (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_INVERTGRN**. This MUST be 0x00000049.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Bounds (16 bytes):** A 128-bit [Windows Metafile Format RectL](#) object, specified in [\[MS-WMF\]](#) section 2.2.1.14, which specifies the bounding rectangle.

**RgnDataSize (4 bytes):** A 32-bit unsigned integer that specifies the size of region data in bytes.

**RgnData (variable):** A **RgnDataSize** length array of bytes that specifies a [RegionData](#) object in logical units.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.4 EMR\_MOVETOEX Record

The Enhanced Metafile Format EMR\_MOVETOEX record specifies coordinates of the new current position in logical units.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Offset																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_MOVETOEX**. This MUST be 0x0000001B.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Offset (8 bytes):** A 64-bit Windows Metafile Format **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies coordinates of the new current position in logical units.

See section [2.3.11](#) for the specification of other State record types.

#### 2.3.11.5 EMR\_PIXELFORMAT Record

The Enhanced Metafile Format EMR\_PIXELFORMAT record specifies the pixel format of the playback device context to the format specified by the data in this record.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
pfd																															
...																															
...																															
...																															
...																															
...																															
...																															
(pfd cont'd for 2 rows)																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_PIXELFORMAT**. This MUST be 0x00000068.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**pfd (40 bytes):** A [PixelFormatDescriptor \(section 2.2.20\)](#) object that specifies pixel format data.

See section [2.3.11](#) for the specification of other State record types.

#### 2.3.11.6 EMR\_RESTOREDC Record

The Enhanced Metafile Format EMR\_RESTOREDC record restores the playback device context to the specified state. The playback device context is restored by popping state information off a stack created by earlier [EMR\\_SAVEDC records \(section 2.3.11\)](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
SavedDC																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_RESTOREDC**. This MUST be 0x00000022.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**SavedDC (4 bytes):** A 32-bit signed integer that specifies the restore state relative to the current state. This MUST be  $\leq 0$  as no absolute level restore is allowed.

The stack can contain the state information for several instances of the playback device context. If the state specified by the specified field is not at the top of the stack, EMR\_RESTOREDC deletes all state information between the top of the stack and the specified instance.

See section [2.3.11](#) for the specification of other State record types.

#### 2.3.11.7 EMR\_SCALEVIEWPORTEXTEX Record

The Enhanced Metafile Format EMR\_SCALEVIEWPORTEXTEX record respecifies the viewport for a device context by using the ratios formed by the specified multiplicands and divisors.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
xNum																															
xDenom																															
yNum																															
yDenom																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SCALEVIEWPORTEXTEX**. This MUST be 0x0000001F.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**xNum (4 bytes):** A 32-bit signed integer that specifies the horizontal multiplicand. Cannot be zero.

**xDenom (4 bytes):** A 32-bit signed integer that specifies the horizontal divisor. Cannot be zero.

**yNum (4 bytes):** A 32-bit signed integer that specifies the vertical multiplicand. Cannot be zero.

**yDenom (4 bytes):** A 32-bit signed integer that specifies the vertical divisor. Cannot be zero.

The extent cannot be changed if the device context is using a fixed scale mapping mode. Only **MM\_ISOTROPIC** and **MM\_ANISOTROPIC** are not fixed scale. The viewport extents are modified as follows:

$$\begin{aligned}x_{\text{NewWE}} &= (x_{\text{OldWE}} * x_{\text{Num}}) / x_{\text{Denom}} \\y_{\text{NewWE}} &= (y_{\text{OldWE}} * y_{\text{Num}}) / y_{\text{Denom}}\end{aligned}$$

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.8 EMR\_SCALEWINDOWEXTEx Record

The Enhanced Metafile Format EMR\_SCALEWINDOWEXTEx record respecifies the window for a playback device context by using the ratios formed by the specified multiplicands and divisors.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Type																															
Size																															
xNum																															
xDenom																															
yNum																															
yDenom																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SCALEWINDOWEXTEx**. This MUST be 0x00000020.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**xNum (4 bytes):** A 32-bit signed integer that specifies the horizontal multiplicand. MUST NOT be zero.

**xDenom (4 bytes):** A 32-bit signed integer that specifies the horizontal divisor. MUST NOT be zero.

**yNum (4 bytes):** A 32-bit signed integer that specifies the vertical multiplicand. MUST NOT be zero.

**yDenom (4 bytes):** A 32-bit signed integer that specifies the vertical divisor. MUST NOT be zero.

The extent cannot be changed if the device context is using a fixed scale mapping mode. Only **MM\_ISOTROPIC** and **MM\_ANISOTROPIC** are not fixed scale. The window extents are modified as follows:

$$\begin{aligned} \text{xNewWE} &= (\text{xOldWE} * \text{xNum}) / \text{xDenom} \\ \text{yNewWE} &= (\text{yOldWE} * \text{yNum}) / \text{yDenom} \end{aligned}$$

See section [2.3.11](#) for the specification of other State record types.

**2.3.11.9 EMR\_SETARCDIRECTION Record**

The Enhanced Metafile Format (EMF) EMR\_SETARCDIRECTION record specifies the drawing direction to be used for arc and rectangle output.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ArcDirection																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETARCDIRECTION**. This MUST be 0x00000039.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ArcDirection (4 bytes):** A 32-bit unsigned integer that specifies the arc direction. The value MUST be in the [ArcDirection \(section 2.1.2\)](#) enumeration.

The default direction is counterclockwise.

The **EMR\_SETARCDIRECTION** record affects the direction in which the following records draw:

- [EMR\\_ARC](#)
- [EMR\\_ARCTO](#)
- [EMR\\_CHORD](#)
- [EMR\\_ELLIPSE](#)
- [EMR\\_PIE](#)
- [EMR\\_RECTANGLE](#)
- [EMR\\_ROUNDRECT](#)

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.10 EMR\_SETBKCOLOR Record

The Enhanced Metafile Format EMR\_SETBKCOLOR record specifies the background color.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Color																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETBKCOLOR**. This MUST be 0x00000019.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Color (4 bytes):** A 32-bit Windows Metafile Format **ColorRef** object, specified in [\[MS-WMF\]](#) section 2.2.1.7, that specifies the background color value.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.11 EMR\_SETBKMODE Record

The Enhanced Metafile Format EMR\_SETBKMODE record specifies the background mix mode of the playback device context. The background mix mode is used with text, hatched brushes, and pen styles that are not solid lines.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
BackgroundMode																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETBKMODE**. This MUST be 0x00000012.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**BackgroundMode (4 bytes):** A 32-bit unsigned integer that specifies the background mode and MUST be in the [BackgroundMode \(section 2.1.4\)](#) enumeration.

See section [2.3.11](#) for the specification of other State record types.



### 2.3.11.12 EMR\_SETBRUSHORGEX Record

The Enhanced Metafile Format EMR\_SETBRUSHORGEX record specifies the origin of the current brush.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Origin																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETBRUSHORGEX**. This MUST be 0x0000000D.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Origin (8 bytes):** A 64-bit Windows Metafile Format **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies the brush horizontal and vertical origin in device units.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.13 EMR\_SETCOLORADJUSTMENT Record

The Enhanced Metafile Format EMR\_SETCOLORADJUSTMENT record specifies color adjustment values for [EMR\\_STRETCHBLT](#) and [EMR\\_STRETCHDIBITS](#) records.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ColorAdjustment																															
...																															
...																															
...																															
...																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETCOLORADJUSTMENT**. This MUST be 0x00000064.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ColorAdjustment (24 bytes):** A 192-bit [ColorAdjustment \(section 2.2.1\)](#) object that specifies the color adjustment values.

Color adjustment values are used to adjust the input color of the source bitmap for EMR\_STRETCHBLT and EMR\_STRETCHDIBITS records when **STRETCH\_HALFTONE** mode is set. The object specified by this record MUST be used in drawing operations that require a ColorAdjustment object, until a different ColorAdjustment object is specified by another EMR\_SETCOLORADJUSTMENT record, or the object is removed by a [EMR\\_DELETEOBJECT](#) record.

See section [2.3.11](#) for the specification of other State record types.

#### 2.3.11.14 EMR\_SETICMMODE Record

The Enhanced Metafile Format (EMF) EMR\_SETICMMODE record specifies Image Color Management (ICM) to be enabled, disabled, or queried on a given device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ICMMode																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETICMMode**. This MUST be 0x00000062.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ICMMode (4 bytes):** A 32-bit unsigned integer that specifies the **ICMMode** to be set. The value MUST be in the [ICMMode \(section 2.1.18\)](#) enumeration. If the system cannot find an **ICC color profile** to match the state of the device, EMR\_SETICMMode MUST fail.

When ICM is enabled, colors specified in most EMF records MUST be **color matched**. The primary exceptions are [EMR\\_BITBLT](#) and [EMR\\_STRETCHBLT](#). The assumption is that when performing a bit-block transfer, the source and destination color profiles are already compatible and need no color correction. However, if a device-independent bitmap (DIB) is used as the source for a bit-block transfer, and the transfer is performed to a destination that has ICM enabled, color matching MUST be performed.

The default color profile for an output destination SHOULD be used when a bit-block transfer is performed to it by the [EMR\\_SETDIBTSTODEVICE](#) or [EMR\\_STRETCHDIBITS](#) record. If this is not what is desired, ICM SHOULD be turned off for the destination before processing either the EMR\_SETDIBTSTODEVICE or EMR\_STRETCHDIBITS record.

See section [2.3.11](#) for the specification of other State record types.

#### 2.3.11.15 EMR\_SETICMPROFILEA Record

The Enhanced Metafile Format EMR\_SETICMPROFILEA record specifies how to set a specified color profile as the output profile for a playback device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
dwValues																															
cbName																															
cbData																															
Data (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETICMPROFILEA**. This MUST be 0x00000070.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**dwValues (4 bytes):** A 32-bit unsigned integer that contains color profile flags.

**cbName (4 bytes):** A 32-bit unsigned integer that specifies the size of desired profile name.

**cbData (4 bytes):** A 32-bit unsigned integer that specifies the size of profile profile data if attached.

**Data (variable):** A byte array of size **cbName+cbData** that specifies the [\[UNICODE\]](#) color profile name and data.

See section [2.3.11](#) for the specification of other State record types.

#### 2.3.11.16 EMR\_SETICMPROFILEW Record

The Enhanced Metafile Format EMR\_SETICMPROFILEW record specifies how to set a specified color profile as the output profile for a playback device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
dwValues																															
cbName																															
cbData																															
Data (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETICMPROFILEW**. This MUST be 0x00000071.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**dwValues (4 bytes):** A 32-bit unsigned integer that contains color profile flags.

**cbName (4 bytes):** A 32-bit unsigned integer that specifies the size of desired profile name.

**cbData (4 bytes):** A 32-bit unsigned integer that specifies the size of color profile data if attached.

**Data (variable):** A byte array of size **cbName** + **cbData** that specifies the [\[UNICODE\]](#) color profile name and data.

See section [2.3.11](#) for the specification of other State record types.

#### 2.3.11.17 EMR\_SETLAYOUT Record

The Enhanced Metafile Format (EMF) EMR\_SETLAYOUT record respecifies the layout of a playback device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
LayoutMode																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETLAYOUT**. This MUST be 0x00000073.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**LayoutMode (4 bytes):** A 32-bit unsigned integer that specifies a value in the [Layout](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.15**.

The layout specifies the order in which text and graphics are revealed in a window or a playback device context. The default is left to right. The **EMR\_SETLAYOUT** record can change this to be right to left, which is the standard in some languages.

Once the layout mode is set to **LAYOUT\_RTL**, values normally specifying right or left are reversed.

See section [2.3.11](#) for the specification of other State record types.

#### 2.3.11.18 EMR\_SETLINKEDUFIS Record

The Enhanced Metafile Format (EMF) EMR\_SETLINKEDUFIS record sets the [UniversalFontIds](#) (section [2.2.24](#)) of the linked fonts to use during character lookup.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
uNumLinkedUFI																															
ufis (variable)																															
...																															
Reserved																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETLINKEDUFIS**. This MUST be 0x00000077.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**uNumLinkedUFI (4 bytes):** A 32-bit unsigned integer specifying the number of UFI's to follow.

**ufis (variable):** An array of **uNumLinkedUFI** elements of type UniversalFontId, which specify the identifiers of the linked fonts.

**Reserved (8 bytes):** This field is reserved and MUST be ignored.

See section [2.3.11](#) for the specification of other State record types.

#### 2.3.11.19 EMR\_SETMAPMODE Record

The Enhanced Metafile Format EMR\_SETMAPMODE record specifies the mapping mode of the playback device context. The mapping mode specifies the unit of measure used to transform page-space units into device-space units, and also specifies the orientation of the device's x and y axes.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
MapMode																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETMAPMODE**. This MUST be 0x00000010.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**MapMode (4 bytes):** A 32-bit unsigned integer whose definition MUST be in the [MapMode \(section 2.1.21\)](#) enumeration.

**MM\_TEXT** mode allows applications to work in device pixels, whose size varies from device to device.

**MM\_HIENGLISH**, **MM\_HIMETRIC**, **MM\_LOENGLISH**, **MM\_LOMETRIC**, and **MM\_TWIPS** modes are useful for applications drawing in physically meaningful units such as inches or millimeters.

**MM\_ISOTROPIC** mode ensures a 1:1 aspect ratio.

**MM\_ANISOTROPIC** mode allows the x-coordinates and y-coordinates to be adjusted independently.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.20 EMR\_SETMAPPERFLAGS Record

The Enhanced Metafile Format EMR\_SETMAPPERFLAGS record respecifies the algorithm the font mapper uses when it maps logical fonts to physical fonts.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Values																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETMAPPERFLAGS**. This MUST be 0x00000010.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.



**Values (4 bytes):** A 32-bit unsigned integer that specifies whether the font mapper SHOULD attempt to match a font's aspect ratio to the current device's aspect ratio. If bit zero is set, the mapper selects only matching fonts.

See section [2.3.11](#) for the specification of other State record types.

**2.3.11.21 EMR\_SETMITERLIMIT Record**

The Enhanced Metafile Format EMR\_SETMITERLIMIT record specifies the limit for the length of miter joins for the playback device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
MiterLimit																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETMITERLIMIT**. This MUST be 0x0000003A.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**MiterLimit (4 bytes):** A 32-bit float that specifies the new miter limit.

See section [2.3.11](#) for the specification of other State record types.

**2.3.11.22 EMR\_SETPOLYFILLMODE Record**

The Enhanced Metafile Format EMR\_SETPOLYFILLMODE record defines polygon fill mode.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
PolygonFillMode																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETPOLYFILLMODE**. This MUST be 0x00000013.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**PolygonFillMode (4 bytes):** A 32-bit unsigned integer that specifies the polygon fill mode and MUST be in the [PolygonFillMode \(section 2.1.27\)](#) enumeration.

In general, the modes differ only in cases where a complex, overlapping polygon **MUST** be filled; for example, a five-sided polygon that forms a five-pointed star with a pentagon in the center. In such cases, **ALTERNATE** mode **SHOULD** fill every other enclosed region within the polygon (the points of the star), but **WINDING** mode **SHOULD** fill all regions (the points of the star and the pentagon).

When the fill mode is **ALTERNATE**, the area between odd-numbered and even-numbered polygon sides on each scan line **SHOULD** be filled. That is, the area between the first and second side **SHOULD** be filled, and between the third and fourth side, and so on.

When the fill mode is **WINDING**, any region that has a non-zero winding value **SHOULD** be filled. The winding value is the number of times a pen used to draw the polygon would go around the region. The direction of each edge of the polygon is significant.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.23 EMR\_SETROP2 Record

The Enhanced Metafile Format EMR\_SETROP2 record defines a binary raster operation mode.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
ROP2Mode																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETROP2**. This **MUST** be 0x00000014.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**ROP2Mode (4 bytes):** A 32-bit unsigned integer that specifies the raster operation mode and **MUST** be in the Windows Metafile Format **Binary Raster Operation** enumeration, specified in [\[MS-WMF\]](#) section **2.1.2**.

Binary raster operation mix modes define how to combine source and destination colors when drawing with the current pen. The mix modes are binary raster operation codes, representing all possible Boolean functions of two variables, using the binary operations AND, OR, and XOR (exclusive OR), and the unary operation NOT. The mix mode is for raster devices only; it is not available for vector devices.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.24 EMR\_SETSTRETCHBLTMODE Record

The Enhanced Metafile Format EMR\_SETSTRETCHBLTMODE record specifies bitmap stretch mode.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
StretchMode																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETSTRETCHBLTMODE**. This MUST be 0x00000015.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**StretchMode (4 bytes):** A 32-bit unsigned integer that specifies the stretch mode and MAY be in the [StretchMode](#) enumeration.

The stretching mode specifies how to combine rows or columns of a bitmap with existing pixels on the display device the [EMR\\_STRETCHBLT](#) record is processed.

The **STRETCH\_ANDSCANS** and **STRETCH\_ORSCANS** modes are typically used to preserve foreground pixels in monochrome bitmaps. The **STRETCH\_DELETESCANS** mode is typically used to preserve color in color bitmaps.

The **STRETCH\_HALFTONE** mode is slower and requires more processing of the source image than the other three modes; but produces higher quality images. Also note that an [EMR\\_SETBRUSHORGE](#) SHOULD be encountered after setting the **STRETCH\_HALFTONE** mode to avoid brush misalignment.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.25 EMR\_SETTEXTALIGN Record

The Enhanced Metafile Format EMR\_SETTEXTALIGN record specifies text alignment.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
TextAlignmentMode																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETTEXTALIGN**. This MUST be 0x00000016.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**TextAlignmentMode (4 bytes):** A 32-bit unsigned integer that specifies text alignment by using a mask of the values in the [TextAlignmentMode](#) enumeration. Only one value can be chosen from those that affect horizontal and vertical alignment. In addition, only one of the two values that alter the current position SHOULD be chosen.

When the current font has a vertical default base line, as with Kanji, the values from the **VerticalTextAlignmentMode** enumeration MUST be used instead of **TA\_BASELINE** and **TA\_CENTER**.

The [EMR\\_SMALLTEXTOUT](#), [EMR\\_EXTTEXTOUTA](#) and [EMR\\_EXTTEXTOUTW](#) records use text-alignment values to position a string of text on the output medium. The values specify the relationship between a reference point and a rectangle that bounds the text. The reference point is either the current position or a point passed to a text output record.

The rectangle that bounds the text is formed by the character cells in the text string.

See section [2.3.11](#) for the specification of other State record types.

**2.3.11.26 EMR\_SETTEXTCOLOR Record**

The Enhanced Metafile Format MR\_SETTEXTCOLOR record defines the current text color.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Color																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETTEXTCOLOR**. This MUST be 0x00000018.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Color (4 bytes):** A 32-bit Windows Metafile Format **ColorRef** object, specified in [\[MS-WMF\]](#) section 2.2.1.7, that specifies the text color value.

See section [2.3.11](#) for the specification of other State record types.

**2.3.11.27 EMR\_SETTEXTJUSTIFICATION Record**

The Enhanced Metafile FormatEMR\_SETTEXTJUSTIFICATION record sets the amount of extra space to add to break characters for justification purposes.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
nBreakExtra																															
nBreakCount																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETTEXTJUSTIFICATION**. This MUST be 0x00000078.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**nBreakExtra (4 bytes):** A 32-bit signed integer that specifies the total amount of extra space, in logical units, to add.

**nBreakCount (4 bytes):** A 32-bit signed integer that specifies the number of break characters.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.28 EMR\_SETVIEWPORTEXTEX Record

The Enhanced Metafile Format EMR\_SETVIEWPORTEXTEX record defines the viewport extent.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Extent																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETVIEWPORTEXTEX**. This MUST be 0x0000000B.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Extent (8 bytes):** A 64-bit Windows Metafile Format **SizeL** object, specified in [\[MS-WMF\]](#) section 2.2.1.16, that specifies the horizontal and vertical extents in device units.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.29 EMR\_SETVIEWPORTORGEX Record

The Enhanced Metafile Format EMR\_SETVIEWPORTORGEX record defines the viewport origin.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Origin																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETVIEWPORTORGEX**. This MUST be 0x0000000C.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Origin (8 bytes):** A 64-bit Windows Metafile Format **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies the window horizontal and vertical origin in device units.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.30 EMR\_SETWINDOWEXTEX Record

The Enhanced Metafile Format EMR\_SETWINDOWEXTEX record defines the window extent.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Extent																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETWINDOWEXTEX**. This MUST be 0x00000009.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Extent (8 bytes):** A 64-bit Windows Metafile Format **SizeL** object, specified in [\[MS-WMF\]](#) section 2.2.1.16, that specifies the horizontal and vertical extents in logical units.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.11.31 EMR\_SETWINDOWORGEX Record

The Enhanced Metafile Format EMR\_SETWINDOWORGEX record defines the window origin.

										1										2												3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
Type																																	
Size																																	
Origin																																	
...																																	

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETWINDOWORGEX**. This MUST be 0x0000000A.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Origin (8 bytes):** A 64-bit Windows Metafile Format **PointL** object, specified in [\[MS-WMF\]](#) section 2.2.1.11, that specifies the window horizontal and vertical origin in logical units.

See section [2.3.11](#) for the specification of other State record types.

### 2.3.12 Transform Record Types

The Enhanced Metafile Format (EMF) transform record types specify and modify world space to page space transforms.

The following are EMF transform record types:

Name	Section	Description
EMR_MODIFYWORLDTRANSFORM	<a href="#">2.3.12.1</a>	Modifies the current world space to page space transform.
EMR_SETWORLDTRANSFORM	<a href="#">2.3.12.2</a>	Specifies a two-dimensional linear transform between world space and page space.

The generic structure of EMF transform records is specified as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Xform																															
...																															
...																															
...																															
...																															
dParm (variable)																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that defines the type of record. The transform record types are listed below. See the table above for descriptions of these record types.

Name	Value
EMR_SETWORLDTRANSFORM	0x00000023
EMR_MODIFYWORLDTRANSFORM	0x00000024

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size, in bytes, of the **dParm** field in the record.

**Xform (24 bytes):** A 192-bit [XForm object \(section 2.2.25\)](#) object that defines a world space to page space transform.

**dParm (variable):** An optional array of 32-bit unsigned integers that specify additional parameters for the transform record.

### 2.3.12.1 EMR\_MODIFYWORLDTRANSFORM Record

The Enhanced Metafile Format (EMF) EMR\_MODIFYWORLDTRANSFORM record modifies the current world space to page space transform.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Xform																															
...																															
...																															
...																															
...																															
...																															
ModifyWorldTransformMode																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_MODIFYWORLDTRANSFORM**. This MUST be 0x00000024.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Xform (24 bytes):** A 192-bit [XForm object \(section 2.2.25\)](#) object that defines the world space to page space transform.

**ModifyWorldTransformMode (4 bytes):** A 32-bit unsigned integer that specifies how the transform modifies the current world space to page space transform. The value MUST be in the [ModifyWorldTransformMode enumeration \(section 2.1.24\)](#).

For more information concerning transforms and **coordinate spaces**, see [\[MSDN-WRLDPGSPC\]](#).

See section [2.3.12](#) for the specification of other transform record types.

### 2.3.12.2 EMR\_SETWORLDTRANSFORM Record

The Enhanced Metafile Format (EMF) EMR\_SETWORLDTRANSFORM record specifies a two-dimensional linear transform between world space and page space.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type																															
Size																															
Xform																															
...																															
...																															
...																															
...																															
...																															

**Type (4 bytes):** A 32-bit unsigned integer that identifies this record type as **EMR\_SETWORLDTRANSFORM**. This MUST be 0x00000023.

**Size (4 bytes):** A 32-bit unsigned integer that specifies the size of this record in bytes.

**Xform (24 bytes):** A 192-bit [XForm object \(section 2.2.25\)](#) object that defines the world space to page space transform. This record uses logical units.

For more information concerning transforms and coordinate spaces, see [\[MSDN-WRLDPGSPC\]](#).

See section [2.3.12](#) for the specification of other transform record types.

## 3 Structure Examples

The following sections describe several operations as used in common scenarios to illustrate the function of the Enhanced Metafile Format.

### 3.1 Metafile Design

#### 3.1.1 Managing Objects

##### 3.1.1.1 EMF Object Table

When a graphics object is created by an Enhanced Metafile Format (EMF) record, it is assigned an index, so that the same object can be retrieved when it is needed to process a subsequent EMF metafile record. The EMF Object Table refers to a possible implementation of a mechanism to save and retrieve these graphics objects. It is simply an indexed table of EMF graphics objects that are defined during the processing of an EMF metafile.

The following types of objects **MUST** be managed:

- Brushes
- Color spaces
- Fonts
- Pens
- Palettes

After a graphics object is placed in the EMF Object Table, it may be removed at some point during metafile processing. A [EMR\\_DELETEOBJECT](#) record is used to disassociate an object from its index, so the index can be reused for a new object. The same index **MAY** be reused many times, but during the lifetime of a given object, the object's index **MUST** be unique.

Thus, the recommended approach to managing objects is:

1. Create an object table associated with the metafile. The maximum number of objects that will ever exist simultaneously is specified in the **Handles** field of the [EMR\\_HEADER](#) record, specified in section [2.3.4.2](#). The maximum number of objects value does not include any pre-defined "stock objects" that may be used.
2. As objects are created, each new object **MUST** be assigned the first available index in the object table.
3. When an object is needed for graphics drawing, an [EMR\\_SELECTOBJECT](#) record **MUST** be received for it, and the object table index specified in the record **MUST** match the index that was assigned to the object when it was created.
4. When a [EMR\\_DELETEOBJECT](#) record is processed for an object that was in the metafile, that index in the table **MUST** be marked as open.
5. After an object is deleted, the next object that is created **MUST** again reuse the first available object table index, which may be the one freed by the deletion.

**Note** There are some index values that are reserved:

- The index value 0 is reserved; it refers to the EMF metafile itself.
- Indexes that have the most-significant bit set refer to stock objects, specified in the [StockObject](#) enumeration in section [2.1.31](#).

Object creation, selection, and deletion depend on proper ordering during playback to achieve the expected results.

### 3.1.2 Byte Ordering

The following code snippet illustrates how the use of the big-endian and little-endian methods can affect the compatibility of applications.

```
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
int main()
{
    int buf;
    int in;
    int nread;
    in = open("file.in", O_RDONLY);
    nread = read(in, (int *) &buf, sizeof(buf));
    printf("First Integer in file.in = %x\n", buf);
    exit(0);
}
```

In the preceding code, if the first integer word stored in the file.in file on a big-endian computer was the hexadecimal number 0x12345678, the resulting output on that computer would be as follows:

```
% ./test
First Integer in file.in = 12345678
%
```

If the file.in file were read by the same program running on a little-endian computer, the resulting output would be as follows:

```
% ./test
First Integer in file.in = 78563412
%
```

Because of the difference in output, one would need to implement metafile record processing so that it can read integers from a file based on the endian method that the output computer uses.

Because metafiles were developed and written with little-endian computers, machines that are big-endian based will have to perform this necessary compensation.

### 3.1.3 Run-Length Encoding (RLE) Bitmap Compression

Metafile records can contain compressed bitmaps that define their colors with 8 or 4 bits-per-pixel.

Compression forms part of the following field names in the bitmap information header structures for different platforms.

When the **Compression** field of the bitmap information header structure is **BI\_RLE8**, a run-length encoding (RLE) format is used to compress an 8-bit bitmap. This format can be compressed in encoded or absolute modes. Both modes can occur anywhere in the same bitmap.

Encoded mode consists of two bytes: the first byte specifies the number of consecutive pixels to be drawn using the color index contained in the second byte. In addition, the first byte of the pair can be set to zero to indicate an escape character that denotes the end of a line, the end of a bitmap, or a delta, depending on the value of the second byte. The interpretation of the escape depends on the value of the second byte of the pair, which can be one of the following values.

Value	Meaning
0	End of line.
1	End of bitmap.
2	The 2 bytes following the escape contain unsigned values indicating the horizontal and vertical offsets of the next pixel from the current position.

In absolute mode, the first byte is zero and the second byte is a value in the range 03H through FFH. The second byte represents the number of bytes that follow, each of which contains the color index of a single pixel. When the second byte is two or less, the escape has the same meaning as encoded mode. In absolute mode, each run **MUST** be aligned on a word boundary.

The following example shows the hexadecimal values of an 8-bit compressed bitmap:

```
03 04 05 06 00 03 45 56 67 00 02 78 00 02 05 01
02 78 00 00 09 1E 00 01
```

The bitmap expands as follows (two-digit values represent a color index for a single pixel):

```
04 04 04
06 06 06 06 06
45 56 67
78 78
```

move current position 5 right and 1 down

```
78 78
```

end of line

```
1E 1E 1E 1E 1E 1E 1E 1E 1E
```

end of RLE bitmap

When the **Compression** field is **BI\_RLE4**, the bitmap is compressed by using a run-length encoding format for a 4-bit bitmap, which also uses encoded and absolute modes:

In encoded mode, the first byte of the pair contains the number of pixels to be drawn using the color indexes in the second byte. The second byte contains two color indexes, one in its high-order 4 bits and one in its low-order 4 bits. The first of the pixels is drawn using the color specified by the high-order 4 bits, the second is drawn using the color in the low-order 4 bits, the third is drawn using the color in the high-order 4 bits, and so on, until all the pixels specified by the first byte have been drawn.

In absolute mode, the first byte is zero. The second byte contains the number of color indexes that follow. Subsequent bytes contain color indexes in their high- and low-order 4 bits, one color index for each pixel. In absolute mode, each run **MUST** be aligned on a word boundary. The end-of-line, end-of-bitmap, and delta escapes described for **BI\_RLE8** also apply to **BI\_RLE4** compression.

The following example shows the hexadecimal values of a 4-bit compressed bitmap:

```
03 04 05 06 00 06 45 56 67 00 04 78 00 02 05 01
04 78 00 00 09 1E 00 01
```

The bitmap expands as follows (single-digit values represent a color index for a single pixel):

```
0 4 0
0 6 0 6 0
4 5 5 6 6 7
7 8 7 8
```

move current position 5 right and 1 down

```
7 8 7 8
```

end of line

```
1 E 1 E 1 E 1 E 1
```

end of RLE bitmap

## 3.2 EMF Metafile Example

This section describes an example of an Enhanced Metafile Format (EMF) metafile, which when processed renders the following image:



**Figure 2: EMF Metafile Example**

The contents of this metafile example is shown below in hexadecimal bytes. The far-left column is the byte count; the far-right characters are the interpretation of the bytes in the ANSI Character Set. The sections that follow describe the packets that convey this series of bytes.

[MS-EMF] – v20080207  
Enhanced Metafile Format Specification  
Copyright © 2008 Microsoft Corporation.  
Release: Thursday, February 7, 2008

00000400:5B 0C 45 5C 0A 45 5C 0A 45 5C 0A 00 3F 5F 0E 3F [.E\ .E\ .E\... ? .?  
00000410:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F \_ .? \_ .? \_ .? \_ .? \_  
00000420:0E 3E 5E 0D 3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E .> ^ .? \_ .? \_ .A ^ .A ^ .  
00000430:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 A ^ .A ^ .A ^ .A ^ .A ^ .A  
00000440:5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E ^ .A ^ .A ^ .A ^ .A ^ .A  
00000450:0E 41 5E 0E 40 5D 0D 38 51 0E 90 91 8E AC AC AC .A ^ .@] .8Q. □ 'Ž---  
00000460:C3 C3 C3 C5 C6 C6 B9 BB BC B3 B0 AD BA AD 9E C8 ĀĀĀĀĒĒ'»¼³°-žÈ  
00000470:B4 A0 DB C8 B4 E4 D1 BE C2 B5 A9 82 7E 7A 8C 88 ' ŪĒ'āN¼Āµ@, ~zĒ^  
00000480:84 83 80 7E 98 8F 85 BE AC 9A C9 B6 A3 CC BB A9 „fĒ~□ ...¾-šĒfĒi»©  
00000490:CD C0 B3 AB A3 9B 80 7E 76 78 76 70 91 8F 8D 84 ĩĀ³«Ē >Ē~vxvp'□ □ „  
000004A0:86 80 36 4B 0B 39 50 0B 38 4E 0B 37 4D 0B 38 4F †Ē6K.9P.8N.7M.8O  
000004B0:3F 3C 55 0C 40 5A 0D 42 5D 0D 42 5D 0D 42 5D 0D <U.@Z.B].B].B].  
000004C0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43 C\ .C\ .C\ .C\ .B[ .C  
000004D0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \ .C\ .C\ .C\ .C\ .C\ .C  
000004E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\ .C\ .C\ .C\ .C\ .C  
000004F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C C\ .C\ .C\ .C\ .C\ .C  
00000500:5C 0C 43 5C 0C 43 5C 0C 45 5B 0A 45 5B 0A 45 5C \ .C\ .C\ .E[ .E[ .E\ .  
00000510:0A 45 5C 0A 45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F .E\ .E\... ? .? .?  
00000520:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3E 5E 0D .? .? .? .? .> ^ .  
00000530:3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 ? \_ .? \_ .A ^ .A ^ .A  
00000540:5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E ^ .A ^ .A ^ .A ^ .A ^ .A  
00000550:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E .A ^ .A ^ .A ^ .A ^ .A  
00000560:40 5D 0D 3D 50 1C 94 94 94 A1 A1 A0 C9 CA CA BC @] .=P. " " " ; ĒĒĒ¼  
00000570:BB B8 BB AC 9C 4A D2 BB A4 E0 CF BD E0 D0 » .»-œĀ-"Ō»µāĒ¼āD  
00000580:BF DA C7 B5 DE CB B8 9A 93 8B 7C 76 70 A2 93 83 žŪĒµĒĒ.š" < |vpĒ"f  
00000590:C0 AB 95 C6 B2 9D CA B8 A4 D1 C0 AD E0 D2 C4 E8 Ā«•Ē²□ĒĒ, µNĀ-āŌĀĒ  
000005A0:DC CF CD C3 B5 8A 85 74 72 72 61 83 82 7E 34 43 ŪĒĒĒµš...trrAf, ~4C  
000005B0:15 3C 54 0C 38 50 0B 38 4E 0B 39 50 0B 3D 55 0C <T.8P.8N.9P.=U.  
000005C0:40 5B 0D 42 5D 0D 42 5D 0D 42 5D 0D 43 5C 0C 43 @[.B].B].B].C\ .C  
000005D0:5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43 5C 0C 43 5C \ .C\ .C\ .B[ .C\ .C\ .C  
000005E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\ .C\ .C\ .C\ .C\ .C  
000005F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C C\ .C\ .C\ .C\ .C\ .C  
00000600:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \ .C\ .C\ .C\ .C\ .C\ .C  
00000610:0C 43 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A .C\ .E\ .E\ .E\ .E\ .  
00000620:45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E E\... ? .? .? .?  
00000630:3F 5F 0E 3F 5F 0E 3F 5F 0E 3E 5E 0D 3F 5F 0E 3F ? .? .? .> ^ .? .?  
00000640:5F 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E .A ^ .A ^ .A ^ .A ^ .A  
00000650:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E .A ^ .A ^ .A ^ .A ^ .A  
00000660:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 4B A ^ .A ^ .A ^ .A ^ .@] .K  
00000670:56 33 B8 B8 B8 B7 B7 B7 A1 A2 BC AC 9B CC B2 V3, , , ••• ; Ē¼- > Ē²  
00000680:97 D6 C3 AE E5 D6 C6 E4 D4 C4 E1 D2 C2 D9 C6 B4 --ŌĀŌāŌŌĒāŌĀŌĀŪĒ'  
00000690:D7 C3 AD CE BF AF AD 9A 85 BB A5 8D BD A9 93 C3 xĀ-Ē; -š...»Ū□¼©"Ā  
000006A0:AF 9B C8 B5 A2 CE BD AA DC CE C0 E2 D5 C9 E1 D4 - >ĒµĒĒĒ¼Ē ŪĒĀŌŌĒāŌ  
000006B0:C7 D5 C5 B4 B1 A9 95 91 97 7B 4A 57 2D 39 51 0B ĒŌĀ'±©•'- --{JW-9Q.  
000006C0:38 50 0B 38 50 0B 3A 52 0C 3E 58 0C 41 5C 0D 42 8P.8P.:R.>X.A\ .B  
000006D0:5D 0D 42 5D 0D 42 5D 0D 41 5C 0C 43 5C 0C 43 5C ].B].B].A\ .C\ .C\ .C  
000006E0:0C 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C .C\ .B[ .C\ .C\ .C\ .C  
000006F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C C\ .C\ .C\ .C\ .C\ .C  
00000700:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C \ .C\ .C\ .C\ .C\ .C  
00000710:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5D 0A .C\ .C\ .C\ .C\ .C].  
00000720:45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 00 E\ .E\ .E\ .E\ .E\ .  
00000730:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E ? .? .? .? .? .?  
00000740:5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E .? .@] .? .? .A ^  
00000750:0F 41 5E 0F 41 5E 0F 41 5E 0E 41 5E 0E 40 5D 0D .A ^ .A ^ .A ^ .A ^ .@].  
00000760:40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 @] .@] .A ^ .@] .@] .@  
00000770:5D 0D 40 5D 0D 41 5E 0E 41 5E 0E 61 6C 44 D0 D0 ].@] .A ^ .A ^ .A\ DĒDĒ  
00000780:D0 B0 B0 B0 C8 C3 BE DB C8 B3 DC CA B8 EB DD CF Ē°°°ĒĀ¼ŪĒ³ŪĒ.ēŪĒ  
00000790:E8 D9 CA E5 D6 C7 E4 D6 C8 D9 C8 B5 D2 BF AA DD ēŪĒāŌĒāŌĒĒĒµŌž°āŪ  
000007A0:CC BA C7 B3 9E B6 9F 89 BC A7 92 C1 AD 98 C6 B3 Ē°Ē³žŪŪŪ¼š'Ā-~Ē³  
000007B0:9F CC BA A8 DB CD BF DD D0 C2 DF D3 C6 E0 D3 C5 ŪĒ°°ŪĒĒŪĒĒĒāŌĒāŌ  
000007C0:E2 D4 C4 A7 AA 8E 52 62 2B 37 4D 0B 39 51 0B 3B āŌĒš°āžRB+7M.9Q.;  
000007D0:53 0C 3E 57 0C 40 5A 0D 42 5D 0D 42 5D 0D 42 5D S.>W.@Z.B].B].B].  
000007E0:0D 42 5D 0D 44 5D 0D 43 5C 0C 43 5C 0C 45 5B 0D .B].D].C\ .C\ .E[ .  
000007F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C C\ .C\ .C\ .C\ .C\ .C  
00000800:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \ .C\ .C\ .C\ .C\ .C\ .C



00000810:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\C\C\C\C\C\C\  
00000820:43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 C\C\C\C\E\E\E\E  
00000830:5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F \.E\E\E\E[...?\_?  
00000840:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F .? .? .? .? .?  
00000850:0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0F .@].? .? .A^.A^.  
00000860:41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 A^.@].@].@].@].@  
00000870:5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ] .A^.@].@].@].@].@  
00000880:0D 41 5E 0E 3F 5B 0D 6A 79 47 D1 D2 CE C5 C2 BF .A^.?[.jyGNOiAAz  
00000890:EF E6 DD E7 D8 CA EE E0 D3 ED DF D2 EA DC CE E9 iæYçØÊiaóíBôëÜíé  
000008A0:DB CD E1 D1 C1 D6 C3 B0 D7 C4 B1 DA C9 B8 D5 C3 ūíáÑÁÖÅ°×Ã±ÚÊ, ÕÃ  
000008B0:B1 B8 A3 8C BC A7 91 C0 AC 97 C5 B1 9D CB B9 A5 ±, ££¥\$'À--À±□Ê¹¥  
000008C0:D6 C8 B8 CE BF AE D9 CB BD E1 D4 C7 E3 D4 C6 BC ÔÊ, î¿@ÜÊ¼áÔÇãÔÆ¼  
000008D0:B9 A2 53 63 2C 37 4C 0B 3C 55 0C 3E 58 0C 40 5B ¹çSC, 7L.<U.>X.@[  
000008E0:0D 41 5C 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D .A\B].B].B].B].  
000008F0:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43 C\C\C\C\E[.C\C\C  
00000900:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\C\C\C\C\C\C\  
00000910:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\C\C\C\C\C\C\  
00000920:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\C\C\C\C\C\C\C  
00000930:5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C \.C\E\E\E\E\E\E  
00000940:0A 45 5C 0A 45 5C 0C 00 3F 5F 0E 3F 5F 0E 3F 5F .E\E[...?\_?\_?\_?  
00000950:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D .? .? .? .? .@].  
00000960:3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E 40 5D 0D 40 ? .? .A^.A^.@].@  
00000970:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E ] .@].@].@].@].A^  
00000980:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E .@].@].@].@].A^.  
00000990:40 59 0F 75 84 54 A6 B1 8E DC D9 CE F5 EE E8 E7 @Y.u,,T|±žÜÜîðieç  
000009A0:DA CE E5 D7 C8 EB DD D0 EC DF D2 E4 D5 C4 D8 C8 ūíá×ÊëYðíBôãÖÅØÊ  
000009B0:B5 D9 CA B8 D9 CB BA D9 C8 B6 D9 C8 B6 C4 B0 9C µÜÊ, ÜÊ°ÜÊ¶ÜÊ¶Å°œ  
000009C0:BA A5 90 BF AA 95 C3 AF 9B C9 B6 A3 C9 B9 A7 C7 °¥□¿·\*·Ä~>£¶££¹\$Ç  
000009D0:B7 A6 D1 C2 B3 DC CF C0 E0 D2 C3 CE C4 B3 5B 69 ·{ÑÅ³ŪíÀaôÃîÅ³[i  
000009E0:33 3C 52 0D 40 5A 0D 41 5C 0D 42 5D 0D 42 5D 0D 3<R.@Z.A\B].B].  
000009F0:42 5D 0D 42 5D 0D 41 5C 0C 42 5D 0D 43 5C 0C 43 B].B].A\B].C\C  
00000A00:5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\E[.C\C\C\C  
00000A10:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\C\C\C\C\C\C\  
00000A20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\C\C\C\C\C\C\C  
00000A30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\C\C\C\C\C\C\C  
00000A40:0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A .E\E\E\E\E\E\E\  
00000A50:45 5B 0C 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F E[...? .? .? .?  
00000A60:3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F ?\_?\_?\_?\_@].?\_?\_?  
00000A70:5F 0E 41 5E 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D \_ .A^.A^.@].@].@].  
00000A80:0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D .@].@].@].A^.@].  
00000A90:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 44 5C 12 8B @].@].@].A^D\.<  
00000AA0:97 6A 93 A2 76 C8 CA B5 F5 EE E9 E8 DC CF E0 D0 --j"çvÊÊµðíéëÜíàð  
00000AB0:C1 DF D2 C4 E1 BB A1 D4 AC 8E D5 B2 97 D3 A4 84 ÁBÖÅÁ»|Ö-žž²--Óµ,,  
00000AC0:D3 A5 86 D7 C4 B0 D7 C5 B1 CF BB A6 C2 AE 99 C1 Ó¥†×Å°×Ã±İ|Ä@™Å  
00000AD0:A4 98 C1 AE 99 C5 B0 9B C7 B5 A2 C8 B8 A7 D1 C2 --Ä@™Å°×ÇµçÊ, ŠÑÅ  
00000AE0:B2 E0 D3 C6 E4 D8 CA E2 D5 C6 71 7B 4B 42 58 10 ²àÓÆäØÊäÖÆq{KBX.  
00000AF0:42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 B].B].B].B].B].B  
00000B00:5D 0D 41 5C 0C 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C ] .A\A\C\C\C\C\C  
00000B10:0C 45 5B 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .E[.C\C\C\C\C\C\C\  
00000B20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\C\C\C\C\C\C\C  
00000B30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\C\C\C\C\C\C\C  
00000B40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A .C\C\C\C\C\C\C\E\  
00000B50:45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00 E\E\E\E\E\E\E[..  
00000B60:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F ? .? .? .? .? .?  
00000B70:5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E \_ .?\_@].?\_?\_ .A^  
00000B80:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].  
00000B90:40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].A^.@].@].@  
00000BA0:5D 0D 40 5D 0D 41 5E 0E 4B 60 17 A0 AB 83 92 A2 ] .@].A^K\ .«f'ç  
00000BB0:73 AC B6 94 F4 ED E7 E2 CF BE DA CA BA D7 C4 B2 s-¶"ðíçãîÅÜÊ°×Å²  
00000BC0:D0 86 58 CA 71 3D C9 69 32 C6 5D 24 C4 54 17 CC Ð†XÊq=Êi2Æ] \$ÄT.İ  
00000BD0:86 5D D5 C5 B2 D2 BC A6 C9 B3 9D C2 AD 96 C1 AC †]ÖÅ²Ö¼;Ê³□Ä--Ä-  
00000BE0:97 C8 B1 9A D9 C8 B6 D3 C4 B4 D1 C3 B3 DE D1 C4 --Ê±šÜÊ¶ÓÄ´ÑÅ³ ÞÑÅ  
00000BF0:E0 D3 C5 E3 D5 C6 A3 A4 81 48 5E 13 42 5D 0D 42 áÓÄãÖÆÊ±□H^B].B].  
00000C00:5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 41 5C ] .B].B].B].B].A\  
00000C10:0C 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C .A\C\C\C\C\C\E[.

00000C20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.\C\.\C\.\C\.\C  
00000C30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.\C\.\C\.\C\.\C\.  
00000C40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.\C\.\C\.\C\.\C\.  
00000C50:43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 C\.\C\.\C\.\E\.\E\.  
00000C60:5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F \.E\.\E\.\E[...? .?  
00000C70:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F .? .? .? .? .?  
00000C80:0E 40 5D 0D 3E 5E 0D 3E 5E 0D 40 5D 0D 40 5D 0D .@].>^.>^.@].@].  
00000C90:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@].  
00000CA0:5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].A^.@].@].@].@].  
00000CB0:0D 41 5E 0E 52 64 1F A4 AF 8A 95 A3 75 B3 AC 84 .A^.\RD.ª~Š•fu³~„  
00000CC0:DC AB 87 D7 A3 7D DB CD BD D2 A1 7F CD 76 42 CB Ů«†×£}Ůí¼Ò;íivBĚ  
00000CD0:71 3C C9 69 32 C7 62 2A C5 5B 20 C3 54 19 D8 AD q<Ĥi2ÇB\*Ā[ ĀT.Ø-  
00000CE0:90 E2 D7 CA D3 BE A9 CA B6 A0 D0 BD AA D1 BD A9 □ā×ĖÓ¼©ĖŹ Đ¼ªN¼©  
00000CF0:DD CD BD DD D0 C2 D0 C1 B1 DA CD BE DE D1 C3 DB Ÿí¼ŸĐĀĐĀ±Úí¼ĐNĀŮ  
00000D00:CD BC 91 9A 6E 4E 62 18 42 5D 0D 42 5D 0D 42 5D í¼'šnNB.B].B].B].  
00000D10:0D 42 5D 0D 42 5E 0B 41 5C 0C 41 5C 0C 41 5C 0C .B].B^.\A\.\A\.\A\.  
00000D20:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43 C\.\C\.\C\.\E[.\C\.  
00000D30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.\C\.\C\.\C\.\C\.  
00000D40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.\C\.\C\.\C\.\C\.  
00000D50:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.\C\.\C\.\C\.\C\.  
00000D60:5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C \.\C\.\E\.\E\.\E\.  
00000D70:0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F 5F 0E 3F 5F .E\.\E[...? .? .?  
00000D80:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D .? .? .? .? .@].  
00000D90:3D 5D 0D 3E 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 >].>].@].@].@].@].  
00000DA0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@].  
00000DB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0E .@].@].@].@].@].  
00000DC0:58 67 26 A4 B1 8B B8 B1 8A D6 9F 74 D5 98 6B D7 Xg&ª±< ±ŠÖŸtŌ~k×  
00000DD0:A4 7F DD CF BF D2 90 66 CD 77 43 CA 6E 38 C8 65 ¢Ÿí¿Ö□FíwCĖN8ĖE  
00000DE0:2C C6 60 27 C6 5B 21 C3 56 1B CB 71 40 DD C8 B6 ,E`'E[!ĀV.ĖqĖŸĖŹ  
00000DF0:D5 C3 AF E1 D2 C2 DF D0 C1 D9 C8 B7 E1 D3 C4 E4 ŌĀ~áoĀßĐĀŮĖ•áoĀā  
00000E00:D8 CA D1 C2 B2 E0 D4 C7 E7 DA CD AD AA 90 6F 7F ØĖNĀ²àoÇçÚí~ª□o  
00000E10:47 52 63 1F 41 5B 0D 41 5C 0C 41 5C 0C 42 5D 0D GRC.A[.\A\.\A\].  
00000E20:42 5D 0B 41 5C 0C 41 5C 0C 41 5B 0C 42 5B 0C 43 B].A\.\A\.\A[.B[.C  
00000E30:5B 0C 43 5C 0C 43 5B 0B 43 5C 0C 43 5C 0C 43 5B [.C\.\C[.C\.\C\.\C[  
00000E40:0C 43 5B 0C 43 5B 0C 43 5C 0C 43 5C 0C 43 5C 0C .C[.C[.C\.\C\.\C\.  
00000E50:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.\C\.\C\.\C\.\C\.  
00000E60:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.\C\.\C\.\C\.\C\.  
00000E70:0C 44 5C 0A 44 5C 0A 45 5B 0A 45 5B 0A 45 5B 0A .D\.\D\.\E[.E[.E[.  
00000E80:45 5B 0B 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E E[...? .? .? .? .?  
00000E90:3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 ? .? .? .@].@].@].  
00000EA0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@].  
00000EB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E .@].@].@].@].A^.  
00000EC0:40 5D 0D 40 5D 0D 40 5D 0D 42 5C 0E 60 6E 33 A9 @].@].@].B\.`n3©  
00000ED0:BA 91 CD B0 8B D7 A2 78 D6 99 6D D9 A8 86 E2 D7 Ź'í°<×xŌmŮ'†ā×  
00000EE0:CA DE BC A2 C8 78 44 D0 85 57 D2 8B 61 CC 74 42 ĖĐ¼çİxĐĐ...WŌ<AİtB  
00000EF0:CC 76 46 CD 7D 4F CC 76 47 CF 9C 7A D8 C9 B7 DF ívFí}ŌivGĩezØĖ•ß  
00000F00:D0 C0 DF CF BF DD CE BE E4 D7 C9 EB E0 D4 D1 C3 ĐĀßİ¿Ÿİ¼ā×ĖēāŌNĀ  
00000F10:B3 DC CF C1 F1 E6 DA C8 C0 AE 72 80 4F 56 66 24 ³ŮİĀñæŮĖĀ©reOVF\$  
00000F20:42 5B 0E 41 5C 0C 41 5C 0C 42 5D 0D 43 5C 0C 43 B[.\A\.\A\].B[.C\.  
00000F30:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C \.\C\.\C[.BZ.C[.C\.  
00000F40:0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B 0D 43 5B 0D .B[.C\.\C\.\C[.C[.  
00000F50:43 5B 0D 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C[.C\.\C\.\C\.\C\.  
00000F60:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.\C\.\C\.\C\.\C\.  
00000F70:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.\C\.\C\.\C\.\C\.  
00000F80:43 5C 0C 45 5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00 C\.\E[.E[.E[.E\..  
00000F90:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F ?\_?\_?\_?\_?\_?\_?  
00000FA0:5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D .? .@].@].@].@].  
00000FB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].  
00000FC0:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 @].@].@].A^.@].@].  
00000FD0:5D 0D 40 5D 0D 43 5C 10 78 82 4C B9 BF 9C D8 AB ].@].C\.\x,L¹¿æØ«  
00000FE0:84 D6 A3 7A D6 A2 7C DE BE A5 E7 DA CD E8 DA CB „ŌĖzŌç|Đ¼ŸçŮİĖŮĖ  
00000FF0:DA AA 89 E7 D9 CC DB AE 91 C7 63 2B C6 5D 24 C4 Ůª%çŮİŮŮ'çC+Ė] \$Ā  
00001000:56 1B C2 51 15 D1 A2 83 E1 D4 C5 E1 D2 C2 E1 D1 v.ĀQ.NçfáŌĀáŌĀāN  
00001010:C2 D1 C0 AD D1 C2 B1 E2 D6 C9 D3 C5 B5 D4 C6 B7 ĀNĀ-NĀ±āŌĖŌĀµŌĖ•  
00001020:E2 D6 C8 EA DE D2 95 9A 76 5B 6B 2A 44 5C 0F 41 āŮĖĖĖŮ•šv[k\*D\.\A

243 / 301

00001440:C6 CF BD AB D2 C3 B3 C2 B2 A1 C6 B6 A3 CD BA A7 Eİ½«ÖÄ³Ä²;E¶fí°S  
00001450:CD BD A7 8B 97 65 7D 8E 55 74 85 49 5A 66 21 44 İ½S<--E}ŽUt...IZF!D  
00001460:5C 0E 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\.C\.C\.C\  
00001470:0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B .C[.BZ.C[.C\.B[.  
00001480:43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 42 5A 0C 42 C\.C\.C[.BZ.BZ.B  
00001490:5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C [C\.C\.C\.C\.C\  
000014A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\.C\  
000014B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 C\.C\.C\.C\.C\.E  
000014C0:5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F [.E[.E[.E\..?..?  
000014D0:5F 0E 3F 5F 0C 3F 5F 0C 3F 5F 0C 3F 5F 0E 40 5E .?..?..?..?@^  
000014E0:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].  
000014F0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D @].@].@].@].@].@  
00001500:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 43 5D ].@].@].@].@].C]  
00001510:0F 69 71 33 B3 BC 99 CD BA 96 D6 A4 79 A2 9B 72 .iq3³¼¶İ°-ÖHyç>r  
00001520:B3 AE 91 DE D0 C2 D9 C2 AF CE 9D 7C CF 8E 64 CE ³@'ÐÐÄÜÄ¬İ□ |İŽDİ  
00001530:7D 4C D6 AB 8D DD D4 C9 DB D1 C6 DB CF C2 D8 CA }LÖ«□YÖEÜNEÜİÄØE  
00001540:BC CF C1 B2 D1 C4 B6 DB CF C3 D6 C8 BB C6 B6 A5 ¼İÄ²ÑÄ¶ÜİÄÖE»E¶Y  
00001550:CF C2 B4 D4 C8 BC A1 9B 81 69 6C 3C 60 63 2E 6C İÄ'ÖE¼;□il<C.l  
00001560:77 41 7A 88 54 74 83 4B 60 6C 27 45 5C 0E 43 5C wAz^TtfK'l'E\C\  
00001570:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D .C\.C\.C\.C\  
00001580:42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 BZ.C[.C\.B[.C\  
00001590:5C 0C 43 5B 0D 44 5A 0C 44 5A 0C 44 5B 0B 43 5C \.C[.DZ.DZ.D[.C\  
000015A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\.C\  
000015B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\  
000015C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0C 43 5B \.C\.C\.C\  
000015D0:0C 45 5B 0C 43 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60 .E[.C\..?..?..?  
000015E0:0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D .?..?..?..?A^@].  
000015F0:3D 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D @].@].@].@].@].  
00001600:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D ].@].@].@].@].@].  
00001610:0D 40 5D 0D 40 5D 0D 40 5D 0D 45 5E 10 7D 82 48 .@].@].@].E^},H  
00001620:BE C5 A4 B9 B0 8A BA B2 93 D1 C9 B6 DB AC 89 E6 ¼Än¹°Šº²"ÑE¶Ü-¾æ  
00001630:CC B7 E8 D2 C0 D1 85 55 CC 75 41 CA 6C 36 C6 6E İ•èÖÄÑ..UİuAËl6En  
00001640:3B B5 A4 8E D2 C6 B8 E7 D8 C9 E5 D6 C7 E0 D1 C2 ;µžÖE,çØÉäÖçàÑÄ  
00001650:DA CC BD DB CD C0 D3 C6 B7 BE AF A0 D0 C3 B5 E9 Úİ½ÜİÄÖE•¼¬ ÐÄµé  
00001660:DE D2 EA E2 D6 C3 BE A9 8B 89 64 61 66 34 65 6D ÞÖéäÖÄ¾«¼%DAF4Em  
00001670:3E 68 76 45 63 6C 2D 47 5D 0F 43 5C 0C 43 5C 0C >hvEC1-G].C\  
00001680:43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 C\.C\.C\.C[.BZ.C  
00001690:5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B [.C\.B[.C\.C[.  
000016A0:0D 45 5B 0D 45 5B 0C 45 5B 0C 43 5C 0C 42 5B 0B .E[.E[.E[.C\.B[.  
000016B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\.C\.C\.C\  
000016C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\  
000016D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C .C\.C\.C\.C\  
000016E0:43 5D 0A 00 3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C C]..?..?..?..?  
000016F0:3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D ?..?..?..?A^@].@].  
00001700:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D ].@].@].@].@].@].  
00001710:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].  
00001720:40 5D 0D 40 5D 0D 47 5E 12 91 93 61 BC C2 A1 C3 @].@].G^.'"A¼Ä;Ä  
00001730:C4 AB E8 E0 D4 DF B4 93 D4 96 68 DE B2 93 E0 BB Ä«èàÖß'Ö-hP²"a»  
00001740:A0 CF 81 50 CD 7A 48 CB 71 3C C8 63 2A D3 9A 75 İ□ PİzHËq<EC\*Ósu  
00001750:E1 D5 C7 DB CB BB D8 C7 B7 D7 C6 B5 D8 C7 B6 D8 áÖÇÜE»ØÇ•×ÆµØÇ¶Ø  
00001760:C7 B6 D3 C1 AF C5 B2 9E DB CD BE E9 DE D3 EF E5 Ç¶ÓÄ-Ä²ŽÜİ¼EÞÓİÄ  
00001770:DB F6 EE E4 FA F0 E7 E3 D8 CA BB B0 9A A1 9D 83 ÜöİäüöçäØE°s;□ f  
00001780:5F 74 3B 4A 5E 12 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 qt;J^C\  
00001790:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C \.C\.C[.BZ.C[.C\  
000017A0:0C 42 5B 0B 43 5B 0C 43 5B 0C 43 5C 0C 45 5B 0C .B[.C[.C[.C\  
000017B0:44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 DZ.DZ.B[.B[.C\  
000017C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.C\  
000017D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.C\.C\  
000017E0:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0A 00 C\.C\.C\  
000017F0:3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C 3F 60 0C 3F ?..?..?..?..?  
00001800:5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D .A^@].@].@].@].  
00001810:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].  
00001820:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D @].@].@].@].@].@  
00001830:5D 0D 4E 60 17 9B 9C 6F BB C2 9E D7 D5 C2 E0 C0 ].N`.»eo»Äž×ÖÄàÄ  
00001840:A6 D7 A0 76 D6 9C 72 D8 A1 7A D4 95 6A D0 82 51 ;× vÖæxØ;zÖ•jÐ,Q

00001850:CD 7A 47 CA 6D 38 D7 A7 88 DB BF AA D9 C9 B8 D3 ízGÊm8×S^Ů¿ªÜÉ,Ó  
00001860:C3 B2 D0 BF AC D0 BE AC D1 C0 AE CF BD AA C6 B3 Ā²D¿-Ð¾-ÑÀ@İ¼ªE³  
00001870:9F D1 BF AD E2 D5 C7 E6 DC D0 EA DF D4 E9 DE D3 ŸŇ¿-āŌÇæÜÐÊßÓéPÓ  
00001880:E4 D8 CC DC CE C0 D1 C0 AF CC BD AA 73 79 3E 50 äŌİŮİĀŇĀ İ¼ªsy>P  
00001890:5F 17 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C .C\C\C\C\C\C\C\  
000018A0:0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B .C[.BZ.C[.C\B[.  
000018B0:43 5B 0D 43 5B 0D 43 5C 0C 45 5B 0C 44 5A 0B 44 C[.C[.C\E[.DZ.D  
000018C0:5A 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C Z.B[.B[.C\C\C\  
000018D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\C\C\C\C\C\C\  
000018E0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\C\C\C\C\C\C\C  
000018F0:5C 0C 43 5C 0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F \.C\E[.E\..? .?  
00001900:5F 0E 3F 60 0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E .?`..?`..?`..?` .A^  
00001910:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].  
00001920:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@  
00001930:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 57 64 ].@].@].@].@].WD  
00001940:1E A5 A5 7C B8 BE 9B D4 D1 BF E2 C5 AD D8 A2 78 .¥¥|,¼¿ŌŇ¿ĀĀ-Ōçx  
00001950:D7 9E 74 D8 A4 7E D2 8C 5E D0 83 53 CC 74 40 D1 ×žtøª~ŌE^Ðfsİt@Ň  
00001960:8D 62 DE D0 C2 DC CE C0 D4 C2 B1 CF BD AB D1 BF □BÐĀŮİĀŌĀ±İ¼«Ň¿  
00001970:AD D4 C3 B2 D4 C3 B1 CF BE AB D1 BF AD DC CC BC -ŌĀ²ŌĀ±İ¼«Ň¿-Ůİ¼  
00001980:DE D0 C3 DF D3 C6 DD D0 C3 DC CF C1 DF D3 C5 DA ÐĀßŌÉŸÐĀŮİĀßŌĀŮ  
00001990:CC BF D3 C5 B6 90 8D 58 60 6D 27 56 64 1E 43 5C İ¿ŌĀŹ□□X`m'VD.C\  
000019A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D .C\C\C\C\C\C\C[.  
000019B0:42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B 43 5B 0D 43 BZ.C[.C\B[.C[.C  
000019C0:5B 0D 43 5C 0C 44 5A 0B 44 5A 0B 44 5A 0B 42 5B [.C\DZ.DZ.DZ.B[  
000019D0:0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .B[.C\C\C\C\C\C\  
000019E0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\C\C\C\C\C\C\C  
000019F0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\C\C\C\C\C\C\C  
00001A00:0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60 .E[.E\..? .?`..?  
00001A10:0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D .?`..?`..?` .A^.@].  
00001A20:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@  
00001A30:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@].  
00001A40:0D 40 5D 0D 40 5D 0D 40 5D 0D 62 6B 26 B0 B1 8C .@].@].@].Bk&±E  
00001A50:BB BE A0 E3 DD D0 E8 D7 C8 D8 A2 7A D7 A4 7D D5 »¼ äŸÐè×ÈŌçz×ª}Ō  
00001A60:9C 74 D1 87 57 CE 7C 4A D0 88 5C D7 C2 AF D4 C5 ætŇ+Wİ|JÐ^×Ā-ŌĀ  
00001A70:B5 D5 C4 B2 D9 C9 B8 DD CD BD DC CC BC DA C9 B9 µŌĀ²ÜÉ,Ÿİ¼Ůİ¼ÜÉ¹  
00001A80:D7 C6 B6 D7 C6 B5 DF D0 C0 D7 C7 B6 D7 C8 B9 D5 ×EŹ×EµßÐĀ×ÇŹ×È¹Ō  
00001A90:C7 B9 D9 CB BD E5 DA CD E4 D9 CC E0 D3 C6 D7 CD Ç¹ÜÈ¼äŮİāŮİāŌÉ×İ  
00001AA0:BD 7C 84 48 5B 6A 1F 5C 67 21 43 5C 0C 43 5C 0C ½|„H[j.\g!C\C\  
00001AB0:43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 C\C\C\C[C[.BZ.C  
00001AC0:5B 0D 43 5C 0C 42 5B 0B 43 5B 0D 42 5A 0C 42 5B [.C\B[.C[.BZ.B[  
00001AD0:0B 44 5A 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B .DZ.DZ.DZ.B[.B[.  
00001AE0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C\C\C\C\C\C\C\C  
00001AF0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\C\C\C\C\C\C\C  
00001B00:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C .C\C\C\C\C\C\E[.  
00001B10:45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C E\..? .?`..?`..?  
00001B20:3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 ?`..?` .A^.@].@].@  
00001B30:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@].  
00001B40:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].  
00001B50:40 5D 0D 41 5D 0D 6D 73 32 B9 BA 9B D6 D1 BF EC @].A].ms2¹º¿ŌŇ¿i  
00001B60:E3 DA EE E7 DF E1 C0 A5 D7 AB 88 D7 A3 7E DA A7 äŮİçßĀĀŸ×«^×f~ŮŖ  
00001B70:84 DD B1 94 D9 C4 B1 D4 C5 B4 DB CB BB E9 DB CD „Ÿ±"ŮĀ±ŌĀ²ŮÉ»éŮİ  
00001B80:E9 DB CE E3 D5 C6 DE CF BF DC CD BC DF CF C0 E1 éŮİāŌÉŸİ¿Ůİ¼ßİĀĀ  
00001B90:D1 C2 DF CF C0 DA CA B9 D6 C7 B8 D5 C7 B9 E8 DC ŇĀŖİĀŮÉ¹ŌÇ,ŌÇ¹èŮ  
00001BA0:D1 E8 DC D1 E4 D8 CC E5 D9 CC B2 AD 93 7D 89 53 ŇèŮŇāŌİāŮİ²-"}%s.  
00001BB0:6A 77 32 60 6A 24 44 5C 0C 43 5C 0C 43 5C 0C 43 jw2`j\$D\C\C\C\C  
00001BC0:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C \.C\C[C[.BZ.C[.C\  
00001BD0:0C 42 5B 0B 42 5A 0C 42 5A 0C 42 5B 0B 44 5A 0B .B[.BZ.BZ.B[.DZ.  
00001BE0:44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 DZ.DZ.B[.B[.C\C\C  
00001BF0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\C\C\C\C\C\C\C  
00001C00:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\C\C\C\C\C\C\C  
00001C10:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 45 5C 0A 00 C\C\C\C\C\E[.E\..  
00001C20:40 5D 0D 41 5E 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F @].A^.?\_.?\_.?\_?  
00001C30:5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D \_A^.@].@].@].@].  
00001C40:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .A^.@].@].@].@].  
00001C50:41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 A^.@].@].@].@].A

00001C60:5E 0F 77 79 3C CC CA B4 EB E2 D9 ED E4 DB EF E7 ^.wy<iÊ'eaûiaûiç  
00001C70:E0 EF E7 DE DE CC BB DD CD BC E8 DB CD E9 DC CF àiçBÞi>Ýí«eûíéûí  
00001C80:DC CD BD E5 D6 C7 F1 E4 D8 F0 E3 D6 EA DC CE E5 Ûí«áôçñãððãôéûíá  
00001C90:D6 C7 E1 D2 C2 DF CF C0 DF CF C0 DD CD BD D9 C8 ôçáôâßíâßíâÝí«ûË  
00001CA0:B8 D7 C6 B5 E0 D3 C6 E6 DB CF EB E0 D5 E1 D5 C8 ,×EµaôEæûíëãôãôË  
00001CB0:DF D3 C7 E4 D8 CB A7 A3 88 77 82 4F 77 86 4B 6B ßôçãðË\$^w,Ow+Kk  
00001CC0:71 30 44 5D 0E 41 5C 0C 43 5C 0C 43 5C 0C 43 5C q0D].A\.\C\.\C\  
00001CD0:0C 43 5C 0C 43 5C 0C 43 5B 0D 43 5B 0D 43 5B 0D .C\.\C\.\C\.[.C\.[.  
00001CE0:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A 0B 44 B[.B[.B[.B[.DZ.D  
00001CF0:5A 0B 42 5B 0B 42 5B 0B 44 5A 0B 43 5C 0C 43 5C Z.B[.B[.DZ.C\.\C\  
00001D00:0C 43 5D 0A 43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C .C].C].C].C\.\C\  
00001D10:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C C\.\C\.\C\.\C\.\C  
00001D20:5C 0C 43 5C 0C 42 5B 0B 44 5B 09 00 40 5D 0D 41 \.C\.\B[.D[.D[.A  
00001D30:5E 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 41 5E ^.?\_.?\_.?\_.?\_.A^  
00001D40:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E .@].@].@].@].A^.  
00001D50:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D @].@].@].@].@].@].  
00001D60:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 47 5F 13 80 7F ].@].@].@].G .E  
00001D70:46 DE DA C9 EF E6 DD EE E5 DD F0 E8 E0 F1 E8 E0 FÞÚÉíæÝíáÝðëàñèà  
00001D80:E2 D2 C3 DE CD BD E3 D3 C3 E3 D4 C5 E2 D3 C5 E6 áôÃÞí«áôÃáôÃáôÃá  
00001D90:D9 CA E5 D7 C9 E7 DA CB E9 DA CC E5 D6 C8 E2 D3 ÛËá×ÉçÚËéúíáôËáô  
00001DA0:C4 E0 D1 C2 DA CA B9 D3 C2 B1 CF BD AA CC B9 A6 ÅãÑÅÚË¹ÓÃ±í«¹ì¹;  
00001DB0:D6 C7 B8 E1 D5 C9 E7 DB CF DF D3 C6 E0 D3 C7 DE Ôç.áôËçúíßôËáôçÞ  
00001DC0:D1 C5 D4 C9 B9 7F 85 5B 75 83 4A 74 78 3A 4A 5F ÑÃÓË¹...[ufJtx:J  
00001DD0:14 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .A\.\C\.\C\.\C\.\C\  
00001DE0:42 5B 0B 42 5B 0C 42 5B 0C 43 5B 0C 42 5B 0B 42 B[.B[.B[.C[.B[.B  
00001DF0:5B 0B 42 5B 0B 42 5B 0B 43 5A 0B 43 5A 0B 42 5B [.B[.B[.CZ.CZ.B[  
00001E00:0B 42 5B 0B 44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A .B[.DZ.C\.\C\.\C].  
00001E10:43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 C].C].C\.\C\.\C  
00001E20:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C \.C\.\C\.\C\.\C\.  
00001E30:0C 42 5B 0B 44 5B 09 00 40 5D 0D 41 5E 0E 3F 5F .B[.D[.D[.A^.?.  
00001E40:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 41 5E 0E 40 5D 0D .? .? .? .A^.@].  
00001E50:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 @].@].@].A^.@].@].  
00001E60:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D ].@].@].@].@].@].  
00001E70:0D 40 5D 0D 40 5D 0D 4B 61 15 8E 8B 57 CF CE B6 .@].@].KA.Ž.wíîŧ  
00001E80:EA E2 D6 F0 E7 DF F1 E9 E1 F1 E8 E0 E4 D5 C5 E1 éãôðçßñéãñèããôÃá  
00001E90:D1 C0 E0 D0 C0 E0 D0 C2 E4 D5 C6 E3 D4 C5 E3 D4 ÑÃãÐÃãÐÃãôËãôÃãô  
00001EA0:C5 E5 D6 C8 E8 DB CD EB DD D0 EB DD CF E8 DA CC ÅãôËëûíëÝðëÝíëûí  
00001EB0:E0 D1 C2 D7 C6 B5 CE BB A9 C9 B5 A2 D3 C4 B5 D3 àÑÃ×Eµí«©EµçÓÃµÓ  
00001EC0:C4 B6 DA CD BF E1 D5 C9 E3 D7 CB DD D1 C4 EA DF ÅŧÚÍçáôËã×ÉÝÑãëß  
00001ED0:D4 B7 B4 9C 77 85 4D 76 7D 3E 4F 61 17 41 5C 0C Ô·'æw...Mv]>OA.A\  
00001EE0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 42 C\.\C\.\C\.\C\.\B[.B  
00001EF0:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B [.B[.B[.B[.B[.B[  
00001F00:0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5A 0B 43 5A 0B .B[.B[.B[.CZ.CZ.  
00001F10:44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A 43 5D 0A 43 DZ.C\.\C\.\C].C  
00001F20:5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C ].C\.\C\.\C\.\C\.  
00001F30:0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43 5A 0B .C\.\C\.\C\.\B[.CZ.  
00001F40:44 5B 09 00 40 5D 0D 40 5D 0D 3F 5F 0E 3F 5F 0E D[.D[.D[.A^.@].@].  
00001F50:3F 5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 ? .? .@].@].@].@].  
00001F60:5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D ].@].A^.@].@].@].  
00001F70:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .@].@].@].@].@].  
00001F80:40 5D 0D 50 63 1A A3 9F 71 CA C9 B1 E0 D9 CB F1 @].PC.£ÝqËË±ãÛËñ  
00001F90:E8 E0 F2 EA E2 F2 E9 E0 E7 D8 C8 E4 D4 C4 E1 D0 éãðëãðëãçðËãôÃãð  
00001FA0:C1 E4 D5 CE DD CD BD D7 C6 B6 DE CF BF E5 D6 C8 ÅãôËÝí«×EŧÞíçáôË  
00001FB0:EB DE D0 F0 E3 D6 F1 E5 D8 F0 E3 D6 ED DF D3 E9 ëÞððãðñãððãðíßóé  
00001FC0:DB CD E8 DA CC E3 D4 C5 E3 D7 CA E4 D8 CC D9 CB ûíëúíãôÃã×ËãðíÛË  
00001FD0:BD D9 CB BD E0 D3 C7 DF D3 C6 E7 DC D0 E1 D8 C8 «ÛË«áôçßóËçÛðãðË  
00001FE0:80 8D 56 7A 82 45 55 64 1B 41 5C 0C 43 5C 0C 43 E□ VZ,EUD.A.C\.\C  
00001FF0:5C 0C 43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B \.C\.\B[.B[.B[.B[  
00002000:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B .B[.B[.B[.B[.B[.  
00002010:42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B 44 5A 0B 43 B[.B[.DZ.DZ.DZ.C  
00002020:5C 0C 43 5C 0C 43 5D 0A 43 5D 0A 43 5C 0C 43 5C \.C\.\C\.\C\.\C\.  
00002030:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C .C\.\C\.\C\.\C\.  
00002040:43 5C 0C 43 5C 0C 43 5A 0B 44 5A 0B 44 5B 09 00 C\.\C\.\CZ.DZ.D[..  
00002050:40 5D 0D 40 5D 0D 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F @].@].? .? .? .?  
00002060:5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D \_.@].@].@].@].@].

00002070:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D .A^.@].@].@].@].  
00002080:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 58 @].@].@].@].@].X  
00002090:67 20 B2 AF 88 D8 D4 C5 F0 E7 DE F1 E9 E1 F3 EC g ^~ØÃðçPñéáóí  
000020A0:E4 F3 E9 E1 E9 DA CB E6 D6 C7 E4 D4 C4 E6 D8 C9 áóéáéúËæðÇáðÃæðË  
000020B0:DF D0 C0 D5 C4 B3 DB CB BA E5 D7 C9 EB DD D0 EF ßÐÃÖÅ³ÜË°á×ÉéÝÐÿ  
000020C0:E2 D5 F1 E4 D8 F0 E4 D7 ED DF D2 EE E1 D4 E6 D8 äöñäððð×íßöíäðæð  
000020D0:C9 C1 AE 98 CA B9 A8 E8 DC D1 EC E1 D7 E7 DC D0 ÉÁ@~É¹"ëÜÑiá×çÜÐ  
000020E0:DD D0 C3 D4 C6 B7 D8 CB BC AE AC 8B 79 87 4D 7B ÝÐÃÖÆ•ØË¼@~<y†M{  
000020F0:84 4A 5C 67 20 41 5C 0C 43 5C 0C 43 5C 0C 43 5C „J\g A\C\C\C\C\  
00002100:0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B .B[.B[.B[.B[.B[.  
00002110:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B B[.B[.B[.B[.B[.B  
00002120:5B 0B 44 5A 0B 44 5A 0B 44 5A 0B 44 5A 0B 43 5C 0C 43 5C [.DZ.DZ.DZ.C\C\  
00002130:0C 43 5D 0A 43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C .C].C].C].C\C\C\  
00002140:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 C\C\C\C\C\C\C\  
00002150:5B 0B 44 5A 0B 44 5A 0B 44 5B 09 00 40 5D 0D 40 [.DZ.DZ.D[.].@].@  
00002160:5D 0D 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D ].?.?.?.?.@].@  
00002170:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E .@].@].@].@].A^.  
00002180:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@  
00002190:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 64 6C 29 C3 BD ].@].@].@].Dl)Ã¼  
000021A0:F4 EC E3 F2 E9 E0 F3 EA E2 F5 EE E6 F3 EA E1 òíäðéäóéäðíæóéä  
000021B0:EB DC CD E8 D9 CA E5 D5 C6 E4 D4 C5 E3 D3 C3 D9 ëÜíëÜËäðËäðÃäóÃÜ  
000021C0:C8 B8 D3 C1 AF DC CC BC EB DD D0 EF E2 D6 F2 E5 È.ÓÃ~ÜíæýÐíäððä  
000021D0:DA F2 E5 D9 F4 E7 DB EA DC CE C7 B3 A0 B7 A2 8B ÓðäÜðçÜëÜíç³ •ç<  
000021E0:D1 C2 B3 EB E0 D6 E6 DA CF EA E0 D5 EF E4 DB E8 ÑÃ³ëàðæÜíëäÜíäÜë  
000021F0:DC D1 D0 C5 B4 8A 8F 63 7C 8B 51 79 85 4A 64 69 ÜÑÐÃ´Š□C|<Qy...JDi  
00002200:27 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 'A\C\C\C\C\C[B[.  
00002210:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B B[.B[.B[.B[.B[.  
00002220:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A [.B[.B[.B[.B[.DZ  
00002230:0B 44 5A 0B 44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A .DZ.DZ.C\C\C].  
00002240:43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43 C].C].C\C\C\C\C  
00002250:5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 44 5A 0B 44 5A \C\C\C.B[DZ.DZ  
00002260:0B 44 5A 0B 44 5B 09 00 40 5D 0D 40 5D 0D 3F 5E .DZ.D[.].@].@].?^  
00002270:0D 3F 5E 0D 3F 5E 0D 3F 5E 0D 40 5D 0D 40 5D 0D .?^.?^.?^.@].@].  
00002280:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@  
00002290:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@].@  
000022A0:0D 3F 5C 0C 40 5C 0D 72 74 35 CC C5 AC F4 EC E4 .?.@.\.rt5iÃ~ðíä  
000022B0:F1 E9 E0 F6 EE E6 F5 EE E6 F2 E9 DE ED DE CF EB ñéäðíæðíæðéðíðíë  
000022C0:DC CE E3 D3 C3 D8 C7 B4 DB CB BA DF D0 BF D9 C8 ÜíäóÃðÇ³ÜË°ßÐ;ÜË  
000022D0:B7 D9 C8 B7 ED DE D0 F4 E6 D9 F9 ED E0 FD F0 E4 •ÜË•íðððæÜüíäýðä  
000022E0:FF F3 E7 E0 CF BD CA B6 A1 C7 B3 9D F3 E7 DC F4 ýóçãĩ¼Æ¶;Ç³□çÜð  
000022F0:EA E0 E7 DD D0 E7 DC D0 E8 DD D2 EF E5 DB F0 E7 éäçÝðçÜðÉÝÜíäÜððç  
00002300:DD A5 A6 81 7C 8A 50 76 83 47 6F 71 31 43 5C 0E ÝÝ|□ |ŠPvfGoq1Ã.  
00002310:43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 C\B[.B[.B[.B[.B[.  
00002320:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B [.B[.B[.B[.B[.B[.  
00002330:0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B .B[.B[.B[.DZ.DZ.  
00002340:43 5A 0B 42 5B 0B 42 5B 0B 43 5C 0A 43 5C 0A 43 CZ.B[.B[.C\C\C.C  
00002350:5C 0A 43 5C 0C 43 5C 0C 43 5C 0B 43 5C 0B 43 5C \C\C\C\C\C\C\C\  
00002360:0C 43 5B 0C 42 5A 0A 43 59 0A 44 5A 0B 44 5A 0B .C[.BZ.CY.DZ.DZ.  
00002370:43 5B 09 00 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D C[.].@].@].@].@].  
00002380:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 @].@].@].@].@].@  
00002390:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D ].@].@].@].@].@].@  
000023A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C .@].@].@].@].@].?\  
000023B0:43 5E 0F 80 7E 41 D3 CD BA F2 E9 E0 F4 EB E2 F4 C^~e~AÓí°ðéäðéäð  
000023C0:EA E1 F0 E8 DF E5 D9 CD DD CE BE E4 D5 C7 DC CD éäðéßäÜíÝí¼äðÇÜí  
000023D0:BF D9 CA BB D9 CA BB DA CC BD D2 C4 B6 CC BF B2 çÜË»ÜË»Üí¼ðÃ¶íç²  
000023E0:CC C1 B5 C8 BD B2 C3 B9 AF C0 B6 AB BA AE A2 AA íÁµÈ¼²Ã¹~Ã¶«°@çª  
000023F0:9A 8B A6 95 84 BB AD 9E D7 CE C5 D5 CA BF D3 C6 š<|•„»~ž×íÃÖËçÖË  
00002400:B9 DC CF C1 E5 D7 C9 EB DF D2 E6 DC CF 8D 92 67 ¹ÜíÃä×ÉëßðæÜí□'g  
00002410:7A 87 4D 73 82 45 79 79 39 49 5E 11 43 5C 0C 42 z†Ms,ÿyy9I^~.C\B  
00002420:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B [.B[.B[.B[.B[.B[.  
00002430:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B .B[.B[.B[.B[.B[.  
00002440:42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 B[.B[.DZ.DZ.B[.B  
00002450:5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C [.B[.C\C\C\C\C\C\  
00002460:0C 43 5C 0C 43 5D 0A 43 5D 0A 43 5C 0C 44 5A 0B .C\C].C].C\C.DZ.  
00002470:43 59 0A 43 59 0A 44 5A 0B 44 5A 0B 42 5C 09 00 CY.CY.DZ.DZ.B\..

[MS-EMF] – v20080207  
Enhanced Metafile Format Specification  
Copyright © 2008 Microsoft Corporation.  
Release: Thursday, February 7, 2008



249 / 301

[MS-EMF] – v20080207  
Enhanced Metafile Format Specification  
Copyright © 2008 Microsoft Corporation.  
Release: Thursday, February 7, 2008

000030B0:5F 0D 3E 5F 0D 3E 5F 0D 3F 5F 0D 3F 5F 0D 3F 5F .> .> .? .? .?  
000030C0:0D 3F 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D .? .> .> .> .> .>  
000030D0:3E 60 0B 3D 5F 0A 3E 5F 0A 3E 5F 0A 3F 60 0B 3F > \ '= > > > > ? \ ?  
000030E0:60 0B 3F 60 0B 3F 60 0B 3F 60 0B 3F 60 0B 3F 60 \ ? \ ? \ ? \ ? \ ? \ ?  
000030F0:0B 3F 60 0B 3E 60 0B 3E 60 0B 3E 60 0B 3F 60 0B .? \ .> \ .> \ .? \ .  
00003100:3F 60 0B 3F 60 0B 3F 5F 0D 3F 5F 0D 3D 5F 0A 00 ? \ ? \ ? \ ? \ ? \ ?  
00003110:12 00 00 00 0C 00 00 00 01 00 00 00 52 00 00 00 .....R...  
00003120:70 01 00 00 02 00 00 00 F3 FF FF FF 00 00 00 00 p.....óÿÿÿ...  
00003130:4E 0C 00 00 4E 0C 00 00 C8 00 00 00 00 00 00 01 N...N...È.....  
00003140:04 00 00 02 41 00 72 00 69 00 61 00 6C 00 00 00 ....A.r.i.A.l...  
00003150:00 00 00 00 00 00 00 00 00 00 00 00 0C 45 00 00 .....E...  
00003160:BC 16 E8 FE FE 07 00 00 20 00 CC 00 00 00 00 00 00 ¼.èþþ... .İ.....  
00003170:4C 00 00 00 FE 07 00 00 DA 16 01 B8 FF FF FF FF L...þ...Ú...ÿÿÿÿ  
00003180:00 00 00 00 00 00 00 00 F0 F6 13 00 00 00 00 00 .....ðö.....  
00003190:0E 20 05 27 00 00 00 00 28 00 00 00 00 00 00 00 . . '.....(.....  
000031A0:1C 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ./.....  
000031B0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
000031C0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
000031D0:00 00 00 00 00 00 00 00 28 00 00 00 FF FF FF 00 .....(...ÿÿÿ.  
000031E0:1C 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ./.....  
000031F0:58 00 00 00 2C 00 00 00 00 00 00 00 00 00 00 00 X....,.....  
00003200:00 00 80 3F 00 00 00 00 00 00 00 00 00 00 80 3F ..e?.....e?  
00003210:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
00003220:00 C5 35 00 00 00 00 00 28 00 00 00 59 00 00 00 ÐÅ5.....(...Y...  
00003230:2D 00 00 00 01 00 18 00 00 00 00 00 1C 2F 00 00 -...../..  
00003240:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
00003250:10 00 90 01 00 00 00 00 25 00 00 00 00 00 00 00 ..□.....%.....  
00003260:D3 3F EC FE 07 00 00 79 0D 21 11 00 00 00 00 Ó?ipþ...y.!.....  
00003270:10 00 90 01 00 00 00 00 00 00 00 59 00 00 00 ..□.....Y...  
00003280:2D 00 00 00 64 76 00 08 00 00 00 00 25 00 00 00 -...Dv.....%...  
00003290:0C 00 00 00 02 00 00 00 54 00 00 00 A0 00 00 00 .....T...  
000032A0:00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF .....ÿÿÿÿÿÿÿÿ  
000032B0:01 00 00 00 AB 0A 0D 42 00 00 0D 42 12 00 00 00 .....«..B...B...  
000032C0:05 00 00 00 0E 00 00 00 4C 00 00 00 00 00 00 00 .....L.....  
000032D0:00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF .....ÿÿÿÿÿÿÿÿ  
000032E0:68 00 00 00 53 00 69 00 6D 00 70 00 6C 00 65 00 h...S.i.m.p.l.E.  
000032F0:20 00 53 00 61 00 6D 00 70 00 6C 00 65 00 00 00 .S.A.m.p.l.E...  
00003300:09 00 00 00 03 00 00 00 0B 00 00 00 07 00 00 00 .....  
00003310:03 00 00 00 07 00 00 00 04 00 00 00 09 00 00 00 .....  
00003320:07 00 00 00 0B 00 00 00 07 00 00 00 03 00 00 00 .....  
00003330:07 00 00 00 09 00 00 00 52 00 00 00 70 01 00 00 .....R...p...  
00003340:03 00 00 00 F3 FF FF FF 00 00 00 00 4E 0C 00 00 ....óÿÿÿ....N...  
00003350:4E 0C 00 00 C8 00 00 00 00 00 00 00 04 00 00 02 N...È.....  
00003360:4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00 66 00 M.i.C.r.o.s.o.F.  
00003370:74 00 20 00 53 00 61 00 6E 00 73 00 20 00 53 00 t. .S.A.n.s. .S.  
00003380:65 00 72 00 69 00 66 00 00 00 00 00 00 00 00 00 E.r.i.F.....  
00003390:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
000033A0:00 00 00 00 28 00 00 00 59 00 00 00 2D 00 00 00 ....(...Y...-...  
000033B0:01 00 18 00 00 00 00 00 1C 2F 00 00 00 00 00 00 ...../.....  
000033C0:00 00 00 00 00 00 00 00 00 00 00 00 10 00 90 01 .....□.  
000033D0:00 00 00 00 25 00 00 00 00 00 00 00 D3 3F EC FE ....%.....Ó?ip  
000033E0:FE 07 00 00 79 0D 21 11 00 00 00 00 10 00 90 01 þ...y.!.....□.  
000033F0:00 00 00 00 00 00 00 00 59 00 00 00 2D 00 00 00 .....Y...-...  
00003400:64 76 00 08 00 00 00 00 00 00 00 00 08 0D 00 Dv.....  
00003410:0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
00003420:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
00003430:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
00003440:00 00 00 00 92 A0 CD 02 00 00 00 00 CA BE CD 02 ....' í.....Ê¼Í.  
00003450:00 00 00 00 00 00 00 00 00 00 00 00 FF FF 5A FE .....ÿÿZþ  
00003460:00 00 00 00 00 00 00 00 00 00 00 95 F1 53 FE .....•ñSp  
00003470:FE 07 00 00 BE 06 5A FE FE 07 00 00 87 F2 53 FE þ...¼.Zþþ...tòSp  
00003480:FE 07 00 00 C4 04 5A FE FE 07 00 00 79 0D 21 11 þ...Å.Zþþ...y.!.  
00003490:00 00 00 00 01 00 00 00 00 00 00 F0 02 5A FE .....ð.Zþ  
000034A0:64 76 00 08 00 00 00 00 25 00 00 00 0C 00 00 00 Dv.....%.....  
000034B0:03 00 00 00 52 00 00 00 70 01 00 00 04 00 00 00 ....R...p.....

```

000034C0:F2 FF FF FF 00 00 00 00 4E 0C 00 00 4E 0C 00 00 0ÿÿÿ....N...N...
000034D0:C8 00 00 00 00 00 00 00 04 00 00 02 4D 00 69 00 È.....M.i.
000034E0:63 00 72 00 6F 00 73 00 6F 00 66 00 74 00 20 00 C.r.o.s.o.F.t. .
000034F0:53 00 61 00 6E 00 73 00 20 00 53 00 65 00 72 00 S.A.n.s. .S.E.r.
00003500:69 00 66 00 00 00 00 00 00 00 00 00 00 00 00 i.F.....
00003510:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00003520:C8 F1 13 00 00 00 00 00 10 00 90 01 00 00 00 Èñ.....□.....
00003530:20 AC CF 02 00 00 00 00 10 00 00 00 06 00 00 00 ¬İ.....
00003540:06 00 00 00 04 00 00 00 01 00 00 00 01 00 00 00 .....
00003550:01 00 00 00 01 00 00 00 0D 00 00 00 00 00 00 00 .....
00003560:03 00 00 00 08 00 00 00 3B 09 00 00 00 00 00 00 .....;.....
00003570:A0 E8 07 02 00 00 00 00 03 00 00 00 FE 07 00 00 è.....p...
00003580:90 01 00 00 4D 00 69 00 63 00 72 00 6F 00 73 00 □...M.i.C.r.o.s.
00003590:6F 00 66 00 74 00 20 00 53 00 61 00 00 00 73 00 o.F.t. .S.A...s.
000035A0:20 00 53 00 65 00 72 00 69 00 66 00 00 00 00 00 .S.E.r.i.F.....
000035B0:00 00 00 00 00 00 00 FF FF 5A FE 00 00 00 00 .....ÿÿZp....
000035C0:40 02 5A FE FE 07 00 00 9D 04 00 00 00 00 00 00 @.Zpb...□.....
000035D0:FF FF FF FF FF FF FF 01 00 00 00 00 00 00 00 ÿÿÿÿÿÿÿ.....
000035E0:20 AC CF 02 00 00 00 00 00 00 07 02 00 00 00 00 ¬İ.....
000035F0:10 AC CF 02 00 00 00 00 26 36 E3 76 00 00 00 00 ¬İ.....&6äv....
00003600:00 00 07 02 00 00 00 00 01 00 00 00 00 27 00 00 .....!..
00003610:00 00 00 00 00 00 00 00 20 AC CF 02 64 76 00 08 ..... ¬İ.Dv..
00003620:00 00 00 00 25 00 00 00 0C 00 00 00 04 00 00 00 .....%.....
00003630:28 00 00 00 0C 00 00 00 03 00 00 00 52 00 00 00 (.....R...
00003640:70 01 00 00 03 00 00 00 13 00 00 00 00 00 00 00 p.....
00003650:4E 0C 00 00 4E 0C 00 00 C8 00 00 00 00 00 00 00 N...N...È.....
00003660:04 00 00 02 4D 00 69 00 63 00 72 00 6F 00 73 00 ....M.i.C.r.o.s.
00003670:6F 00 66 00 74 00 20 00 53 00 61 00 6E 00 73 00 o.F.t. .S.A.n.s.
00003680:20 00 53 00 65 00 72 00 69 00 66 00 00 00 00 00 .S.E.r.i.F.....
00003690:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000036A0:00 00 00 00 00 73 00 20 00 53 00 65 00 72 00 .....s. .S.E.r.
000036B0:69 00 66 00 00 00 00 00 00 00 00 00 00 00 00 i.F.....
000036C0:FF FF 5A FE 00 00 00 00 40 02 5A FE FE 07 00 00 ÿÿZp....@.Zpb...
000036D0:9D 04 00 00 00 00 00 00 FF FF FF FF FF FF FF FF □.....ÿÿÿÿÿÿÿÿ
000036E0:01 00 00 00 00 00 00 00 20 AC CF 02 00 00 00 00 ..... ¬İ.....
000036F0:00 00 07 02 00 00 00 00 40 02 5A FE FE 07 00 00 .....@.Zpb...
00003700:F3 14 00 00 00 00 00 00 F3 14 0A 1E 00 00 00 00 ó.....ó.....
00003710:94 8A E8 FE FE 07 00 00 04 00 00 00 00 00 00 00 "Šèpb.....
00003720:65 58 53 FE 00 00 00 00 00 00 00 00 00 00 00 00 EXSp.....
00003730:00 F5 13 00 00 00 00 00 03 01 56 E5 89 1A 00 00 .ö.....Vá%...
00003740:55 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00 U.....
00003750:00 00 00 00 FE 07 00 00 79 0D 21 11 00 00 00 00 ....p...y.!.....
00003760:40 02 5A FE 00 00 00 00 26 06 5A FE FE 07 00 00 @.Zp....&.Zpb...
00003770:08 F5 13 00 00 00 00 00 00 F5 13 00 00 00 00 00 .ö.....ö.....
00003780:07 CB 54 FE FE 07 00 00 79 0D 21 11 00 00 00 00 .ËTp...y.!.....
00003790:04 00 00 00 00 00 00 00 24 07 5A FE FE 07 00 00 .....$.Zpb...
000037A0:01 00 00 00 64 76 00 08 00 00 00 00 25 00 00 00 ....Dv.....%...
000037B0:0C 00 00 00 03 00 00 00 25 00 00 00 0C 00 00 00 .....%.....
000037C0:02 00 00 00 28 00 00 00 0C 00 00 00 04 00 00 00 .....(.....
000037D0:28 00 00 00 0C 00 00 00 03 00 00 00 25 00 00 00 (.....%...
000037E0:0C 00 00 00 0D 00 00 80 0E 00 00 00 14 00 00 00 .....e.....
000037F0:00 00 00 00 10 00 00 00 14 00 00 00 .....

```

### 3.2.1 EMR\_HEADER Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_HEADER](#) record, specified in section [2.3.4.2](#).

```

00000000:01 00 00 00 D4 00 00 00 00 00 00 00 00 00 00 00
00000010:59 00 00 00 59 00 00 00 00 00 00 00 00 00 00 00
00000020:42 0C 00 00 41 0C 00 00 20 45 4D 46 00 00 01 00

```

```

00000030:FC 37 00 00 16 00 00 00 05 00 00 00 34 00 00 00
00000040:6C 00 00 00 00 00 00 00 80 07 00 00 B0 04 00 00
00000050:A5 02 00 00 A7 01 00 00 00 00 00 00 00 00 00 00
00000060:00 00 00 00 D5 55 0A 00 A5 75 06 00 53 00 61 00
00000070:6D 00 70 00 6C 00 65 00 20 00 45 00 4D 00 46 00
00000080:20 00 74 00 68 00 61 00 74 00 20 00 68 00 61 00
00000090:73 00 20 00 61 00 20 00 62 00 72 00 75 00 73 00
000000A0:68 00 20 00 66 00 69 00 6C 00 6C 00 2C 00 20 00
000000B0:62 00 69 00 74 00 6D 00 61 00 70 00 2C 00 20 00
000000C0:61 00 6E 00 64 00 20 00 74 00 65 00 78 00 74 00
000000D0:00 00 00 00

```

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Type (0x00000001)																															
Size (0x000000D4)																															
Bounds (0x00000000)																															
... (0x00000000)																															
... (0x00000059)																															
... (0x00000059)																															
Frame (0x00000000)																															
... (0x00000000)																															
... (0x00000C42)																															
... (0x00000C31)																															

**Figure 3: EMR\_HEADER Record Example, Part 1**

**Type:** 0x00000001 identifies the record type as EMR\_HEADER.

**Size:** 0x000000D4 is the record size in bytes.

**Bounds:** 0x00000000, 0x00000000, 0x00000059, 0x00000059 specifies the rectangular inclusive-inclusive bounds in device units of the smallest rectangle that can be drawn around the image stored in the metafile.

**Frame:** 0x00000000, 0x00000000, 0x00000C42, 0x00000C31 specifies the rectangular inclusive-inclusive dimensions, in .01 millimeter units, of a rectangle that surrounds the image stored in the metafile.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Signature (0x464D4520)																															
Version (0x00010000)																															
Byte (0x000037FC)																															
Records (0x00000016)																															
Handles (0x0005)																Reserved (0x0000)															
nDescription (0x00000034)																															
offDescription (0x0000006C)																															
PalEntries (0x00000000)																															

**Figure 4: EMR\_HEADER Record Example, Part 2**

**Signature:** 0x464D4520 is the record signature, which consists of the ASCII string " EMF".

**Version:** 0x00010000 specifies that this EMF metafile must be interoperable with Windows NT operating system technology.

**Bytes:** 0x000037FC specifies the size of the metafile in bytes.

**Records:** 0x00000016 specifies the number of records in the metafile.

**Handles:** 0x0005 specifies the number of [EMF Object Table](#) entries that will need to be defined during the processing of the metafile. These entries correspond to graphics objects that are used in drawing commands. The entries are indexed; the indexes are used to refer indirectly to the graphics objects. Index 0 is reserved for references to the metafile itself.

**Reserved:** 0x0000 is not used.

**Description:** 0x00000034 specifies the number of characters in the array that contains the description of the metafile's contents.

**offDescription:** 0x0000006C specifies the offset from the beginning of this record to the array that contains the description of the metafile's contents.

**PalEntries:** 0x00000000 specifies the number of entries in the metafile palette. The location of the palette is specified in the [EMR\\_EOF](#) record.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Device (0x00000780)																															
... (0x00000780)																															
Millimeters (0x000002A5)																															
... (0x000001A7)																															
cbPixelFormat (0x00000000)																															
offPixelFormat (0x00000000)																															
bOpenGL (0x00000000)																															
MicrometersX (0x000A55D5)																															
MicrometersY (0x000675A5)																															
Description ("Sample EMF that has a brush fill, bitmap, and text")																															

**Figure 5: EMR\_HEADER Record Example, Part 3**

**Device:** 0x00000780, 0x00000780 specifies the size of the reference device in pixels.

**Millimeters:** 0x000002A5, 0x000001A7 specifies the size of the reference device in millimeters.

**cbPixelFormat:** 0x00000000 specifies the size of the [PixelFormatDescriptor \(section 2.2.20\)](#) structure. This value indicates that no pixel format is defined.

**offPixelFormat:** 0x00000000 specifies the offset to the PixelFormatDescriptor in the metafile. In this case, no pixel format structure is present.

**bOpenGL:** 0x00000000 specifies that no OpenGL commands are present in the metafile.

**Micrometers:** 0x000A55D5, 0x000675A5 specifies the horizontal and vertical size of the reference device in micrometers.

**EmfDescription:** "Sample EMF that has a brush fill, bitmap, and text".

### 3.2.2 EMR\_CREATEBRUSHINDIRECT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_CREATEBRUSHINDIRECT](#) record, specified in section [2.3.7.1](#).

```
000000D0:          27 00 00 00 18 00 00 00 01 00 00 00
000000E0:02 00 00 00 52 47 2A 00 03 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000027)																															
Size (0x00000018)																															
ihBrush (0x00000001)																															
LogBrush (0x00000002)																															
... (0x0052472A)																															
... (0x00000003)																															

**Figure 6: EMR\_CREATEBRUSHINDIRECT Record Example Part 1**

**Type:** 0x00000027 identifies this record type as EMR\_CREATEBRUSHINDIRECT. This value MUST be 0x00000027.

**Size:** 0x00000018 specifies the size of this record in bytes.

**ihBrush:** 0x00000001 specifies brush [EMF Object Table \(section 3.1.1.1\)](#) index.

**LogBrush:** A 96-bit length [LogBrushEx \(section 2.2.10\)](#) object that contains brush data.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BrushStyle (0x00000002)																															
Color (0x0052472A)																															
BrushHatch (0x00000003)																															

**Figure 7: EMR\_CREATEBRUSHINDIRECT Record Example Part 2**

**BrushStyle:** 0x00000002 specifies the brush style. The value MUST be an enumeration from Windows Metafile Format **BrushStyle** enumeration, specified in [\[MS-WMF\]](#) section 2.1.4.

**Color:** 0x0052472A is a 32-bit WMF **ColorRef** object, specified in [\[MS-WMF\]](#) section 2.2.1.7, that specifies the brush color value.

**BrushHatch:** 0x00000003 specifies the brush hatch data. The content depends on the value of **BrushStyle**.

### 3.2.3 EMR\_SELECTOBJECT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_SELECTOBJECT](#) record, specified in section [2.3.8.5](#).

```
000000E0:                                25 00 00 00
000000F0:0C 00 00 00 01 00 00 00
```



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000001)																															

**Figure 8: EMR\_SELECTOBJECT Record Example**

**Type:** 0x00000025 identifies this record type as EMR\_SELECTOBJECT. This value MUST be 0x00000025.

**Size:** 0x0000000C specifies the size of this record in bytes.

**ihObject:** 0x00000001 specifies the index of an object in the [EMF Object Table](#).

### 3.2.4 EMR\_BITBLT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_BITBLT](#) record, specified in section [2.3.1.2](#).

```
000000F0:                4C 00 00 00 64 00 00 00
00000100:00 00 00 00 00 00 00 00 59 00 00 00 59 00 00 00
00000110:00 00 00 00 00 00 00 00 5A 00 00 00 5A 00 00 00
00000120:21 00 F0 00 00 00 00 00 00 00 00 00 00 80 3F
00000130:00 00 00 00 00 00 00 00 80 3F 00 00 00 00
00000140:00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000150:00 00 00 00 00 00 00 00 00 00 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x0000004C)																															
Size (0x00000064)																															
Bounds (0x00000000)																															
... (0x00000000)																															
... (0x00000059)																															
... (0x00000059)																															
xDest (0x00000000)																															
yDest (0x00000000)																															
cxDest (0x00000059)																															
cyDest (0x00000059)																															

**Figure 9: EMR\_BITBLT Record Example, Part 1**

**Type:** 0x0000004C identifies this record type as EMR\_BITBLT. This value MUST be 0x0000004C.

**Size:** 0x00000064 specifies the size of this record in bytes.

**Bounds:** 0x00000000, 0x00000000, 0x00000059, 0x00000059 specifies the bounding rectangle in device units.

**xDest:** 0x00000000 specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

**yDest:** 0x00000000 specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

**cxDest:** 0x0000005A specifies the logical width of the destination rectangle.

**cyDest:** 0x0000005A specifies the logical height of the destination rectangle.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BitBltRasterOperation (0x00F00021)																															
xSrc (0x00000000)																															
ySrc (0x00000000)																															
xformSrc (0x3F800000)																															
... (0x00000000)																															
... (0x00000000)																															
... (0x3F800000)																															
... (0x00000000)																															
... (0x00000000)																															

**Figure 10: EMR\_BITBLT Record Example, Part 2**

**BitBltRasterOperation:** 0x00F00021 specifies the raster-operation code. These codes define how the color data of the source rectangle is to be combined with the color data of the destination rectangle to achieve the final color. The value MUST be in the Windows Metafile Format **Ternary Raster Operation** enumeration, specified in [\[MS-WMF\]](#) section **2.1.32**.

**xSrc:** 0x00000000 specifies the logical x-coordinate of the upper-left corner of the source rectangle.

**ySrc:** 0x00000000 specifies the logical y-coordinate of the upper-left corner of the source rectangle.

**xformSrc:** 0x3F800000, 0x00000000, 0x00000000, 0x3F800000, 0x00000000, 0x00000000 specifies the world-space to page-space transformation (for more information, see [\[MSDN-WRLDPGSPC\]](#)) of the source device context.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bkColorSrc (0x00000000)																															
UsageSrc (0x00000000)																															
offBmiSrc (0x00000000)																															
cbBmiSrc (0x00000000)																															
offBitsSrc (0x00000000)																															
cbBitsSrc (0x00000000)																															

**Figure 11: EMR\_BITBLT Record Example, Part 3**

**BkColorSrc:** 0x00000000 specifies the background color (the RGB value) of the source device context.

**UsageSrc:** 0x00000000 specifies the value of the **Colors** field of the WMF **DeviceIndependentBitmap** object, specified in [\[MS-WMF\]](#) section 2.2.2.3. The value MUST be in the [DIBColors \(section 2.1.9\)](#) enumeration.

**offBmiSrc:** 0x00000000 specifies the offset to the source **DeviceIndependentBitmap** object.

**cbBmiSrc:** 0x00000000 specifies the size of the source **DeviceIndependentBitmap** object.

**offBitsSrc:** 0x00000000 specifies the offset to the source bitmap bits.

**cbBitsSrc:** 0x00000000 specifies the size of the source bitmap bits.

### 3.2.5 EMR\_SELECTOBJECT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_SELECTOBJECT](#) record, specified in section [2.3.8.5](#).

```
00000150:                                25 00 00 00
00000160:0C 00 00 00 00 00 00 80
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x80000000 = WHITE BRUSH)																															

**Figure 12: EMR\_SELECTOBJECT Record Example**

**Type:** 0x00000025 identifies this record type as EMR\_SELECTOBJECT. This value MUST be 0x00000025.

**Size:** 0x0000000C specifies the size of this record in bytes.

**ihObject:** 0x80000000 specifies the index of an object in the [EMF Object Table](#).

### 3.2.6 EMR\_BITBLT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_BITBLT](#) record, specified in section [2.3.1.2](#).

```
00000160:                4C 00 00 00 A8 2F 00 00
00000170:00 00 00 00 2D 00 00 00 59 00 00 00 59 00 00 00
00000180:00 00 00 00 2D 00 00 00 5A 00 00 00 2D 00 00 00
00000190:20 00 CC 00 00 00 00 00 00 00 00 00 00 80 3F
000001A0:00 00 00 00 00 00 00 00 00 00 80 3F 00 00 00 00
000001B0:00 00 00 00 FF FF FF 00 00 00 00 00 64 00 00 00
000001C0:28 00 00 00 8C 00 00 00 1C 2F 00 00 28 00 00 00
000001D0:59 00 00 00 2D 00 00 00 01 00 18 00 00 00 00 00
000001E0:1C 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001F0:00 00 00 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E
00000200:3F 5F 0E 3F 5F 0E 3F 5F 0E 3E 5E 0D 3F 5F 0E 3F
00000210:5F 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E
00000220:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E
00000230:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 3D
00000240:59 0C 5A 60 4E AE AE AE BF BF BF C5 C5 C5 C0 C0
00000250:C1 B1 B2 B1 A1 A1 A1 A9 AA AA AE AF B0 A2 A2 A2
00000260:A6 A6 A4 AF AE AD AC AC AC A6 A6 A6 99 99 99 7D
00000270:7D 7D 66 66 65 5A 5A 59 4F 4F 4F 58 58 58 76 76
00000280:76 9E 9E 9E B5 B5 B5 3C 43 2D 32 46 0A 34 4A 0A
00000290:34 49 0A 36 4C 0B 3A 52 0B 3F 59 0C 41 5C 0D 42
000002A0:5D 0D 42 5D 0D 44 5D 0D 44 5D 0D 43 5C 0C 43 5C
000002B0:0C 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C
000002C0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
000002D0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
000002E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000002F0:45 5B 0C 45 5B 0C 45 5C 0A 45 5C 0A 45 5C 0A 00
00000300:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F
00000310:5F 0E 3F 5F 0E 3E 5E 0D 3F 5F 0E 3F 5F 0E 41 5E
00000320:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E
00000330:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41
00000340:5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 39 52 0C 70 73
00000350:6B AD AD AD A8 A8 A8 99 99 98 A6 A6 A6 A6 A7 A7
00000360:A3 A5 A6 A6 A5 A3 AE A5 9D C9 BB AD A1 9B 95 A2
00000370:9F 9C B3 B1 AF B6 B4 B3 A9 A9 A9 97 93 90 8E 87
00000380:81 89 84 7F 81 7E 7A 79 78 76 80 7F 7E 98 98 98
00000390:BC BC BC 6D 70 65 33 47 0A 37 4D 0B 35 4B 0A 35
000003A0:4B 0B 38 4F 0B 3D 56 0C 41 5B 0D 42 5D 0D 42 5D
000003B0:0D 43 5D 0D 44 5D 0D 43 5C 0C 43 5C 0C 43 5C 0C
000003C0:42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43
000003D0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C
000003E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C
000003F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 45
00000400:5B 0C 45 5C 0A 45 5C 0A 45 5C 0A 00 3F 5F 0E 3F
00000410:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F
00000420:0E 3E 5E 0D 3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E
00000430:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41
00000440:5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E
00000450:0E 41 5E 0E 40 5D 0D 38 51 0E 90 91 8E AC AC AC
00000460:C3 C3 C3 C5 C6 C6 B9 BB BC B3 B0 AD BA AD 9E C8
00000470:B4 A0 DB C8 B4 E4 D1 BE C2 B5 A9 82 7E 7A 8C 88
00000480:84 83 80 7E 98 8F 85 BE AC 9A C9 B6 A3 CC BB A9
00000490:CD C0 B3 AB A3 9B 80 7E 76 78 76 70 91 8F 8D 84
000004A0:86 80 36 4B 0B 39 50 0B 38 4E 0B 37 4D 0B 38 4F
```

000004B0:0B 3C 55 0C 40 5A 0D 42 5D 0D 42 5D 0D 42 5D 0D  
000004C0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43  
000004D0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
000004E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
000004F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00000500:5C 0C 43 5C 0C 43 5C 0C 45 5B 0A 45 5B 0A 45 5C  
00000510:0A 45 5C 0A 45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F  
00000520:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3E 5E 0D  
00000530:3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E 41 5E 0E 41  
00000540:5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E  
00000550:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E  
00000560:40 5D 0D 3D 50 1C 94 94 94 A1 A1 A0 C9 CA CA BC  
00000570:BB B8 BB AC 9C C4 AD 94 D2 BB A4 E0 CF BD E0 D0  
00000580:BF DA C7 B5 DE CB B8 9A 93 8B 7C 76 70 A2 93 83  
00000590:C0 AB 95 C6 B2 9D CA B8 A4 D1 C0 AD E0 D2 C4 E8  
000005A0:DC CF CD C3 B5 8A 85 74 72 72 61 83 82 7E 34 43  
000005B0:15 3C 54 0C 38 50 0B 38 4E 0B 39 50 0B 3D 55 0C  
000005C0:40 5B 0D 42 5D 0D 42 5D 0D 42 5D 0D 43 5C 0C 43  
000005D0:5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43 5C 0C 43 5C  
000005E0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
000005F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00000600:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00000610:0C 43 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A  
00000620:45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E  
00000630:3F 5F 0E 3F 5F 0E 3F 5F 0E 3E 5E 0D 3F 5F 0E 3F  
00000640:5F 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E  
00000650:0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E  
00000660:41 5E 0E 41 5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 4B  
00000670:56 33 B8 B8 B8 B7 B7 B7 A1 A1 A2 BC AC 9B CC B2  
00000680:97 D6 C3 AE E5 D6 C6 E4 D4 C4 E1 D2 C2 D9 C6 B4  
00000690:D7 C3 AD CE BF AF AD 9A 85 BB A5 8D BD A9 93 C3  
000006A0:AF 9B C8 B5 A2 CE BD AA DC CE C0 E2 D5 C9 E1 D4  
000006B0:C7 D5 C5 B4 B1 A9 95 91 97 7B 4A 57 2D 39 51 0B  
000006C0:38 50 0B 38 50 0B 3A 52 0C 3E 58 0C 41 5C 0D 42  
000006D0:5D 0D 42 5D 0D 42 5D 0D 41 5C 0C 43 5C 0C 43 5C  
000006E0:0C 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C  
000006F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00000700:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00000710:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5D 0A  
00000720:45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 00  
00000730:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F  
00000740:5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E  
00000750:0F 41 5E 0F 41 5E 0F 41 5E 0E 41 5E 0E 40 5D 0D  
00000760:40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40  
00000770:5D 0D 40 5D 0D 41 5E 0E 41 5E 0E 61 6C 44 D0 D0  
00000780:D0 B0 B0 B0 C8 C3 BE DB C8 B3 DC CA B8 EB DD CF  
00000790:E8 D9 CA E5 D6 C7 E4 D6 C8 D9 C8 B5 D2 BF AA DD  
000007A0:CC BA C7 B3 9E B6 9F 89 BC A7 92 C1 AD 98 C6 B3  
000007B0:9F CC BA A8 DB CD BF DD D0 C2 DF D3 C6 E0 D3 C5  
000007C0:E2 D4 C4 A7 AA 8E 52 62 2B 37 4D 0B 39 51 0B 3B  
000007D0:53 0C 3E 57 0C 40 5A 0D 42 5D 0D 42 5D 0D 42 5D  
000007E0:0D 42 5D 0D 44 5D 0D 43 5C 0C 43 5C 0C 45 5B 0D  
000007F0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00000800:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00000810:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00000820:43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45  
00000830:5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F  
00000840:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F

00000850:0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0F  
00000860:41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00000870:5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00000880:0D 41 5E 0E 3F 5B 0D 6A 79 47 D1 D2 CE C5 C2 BF  
00000890:EF E6 DD E7 D8 CA EE E0 D3 ED DF D2 EA DC CE E9  
000008A0:DB CD E1 D1 C1 D6 C3 B0 D7 C4 B1 DA C9 B8 D5 C3  
000008B0:B1 B8 A3 8C BC A7 91 C0 AC 97 C5 B1 9D CB B9 A5  
000008C0:D6 C8 B8 CE BF AE D9 CB BD E1 D4 C7 E3 D4 C6 BC  
000008D0:B9 A2 53 63 2C 37 4C 0B 3C 55 0C 3E 58 0C 40 5B  
000008E0:0D 41 5C 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D  
000008F0:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43  
00000900:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00000910:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00000920:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00000930:5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C  
00000940:0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F 5F 0E 3F 5F  
00000950:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D  
00000960:3F 5F 0E 3F 5F 0E 41 5E 0E 41 5E 0E 40 5D 0D 40  
00000970:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E  
00000980:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E  
00000990:40 59 0F 75 84 54 A6 B1 8E DC D9 CE F5 EE E8 E7  
000009A0:DA CE E5 D7 C8 EB DD D0 EC DF D2 E4 D5 C4 D8 C8  
000009B0:B5 D9 CA B8 D9 CB BA D9 C8 B6 D9 C8 B6 C4 B0 9C  
000009C0:BA A5 90 BF AA 95 C3 AF 9B C9 B6 A3 C9 B9 A7 C7  
000009D0:B7 A6 D1 C2 B3 DC CF C0 E0 D2 C3 CE C4 B3 5B 69  
000009E0:33 3C 52 0D 40 5A 0D 41 5C 0D 42 5D 0D 42 5D 0D  
000009F0:42 5D 0D 42 5D 0D 41 5C 0C 42 5D 0D 43 5C 0C 43  
00000A00:5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43 5C 0C 43 5C  
00000A10:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00000A20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00000A30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00000A40:0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A  
00000A50:45 5B 0C 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E  
00000A60:3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F  
00000A70:5F 0E 41 5E 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D  
00000A80:0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D  
00000A90:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 44 5C 12 8B  
00000AA0:97 6A 93 A2 76 C8 CA B5 F5 EE E9 E8 DC CF E0 D0  
00000AB0:C1 DF D2 C4 E1 BB A1 D4 AC 8E D5 B2 97 D3 A4 84  
00000AC0:D3 A5 86 D7 C4 B0 D7 C5 B1 CF BB A6 C2 AE 99 C1  
00000AD0:AD 98 C1 AE 99 C5 B0 9B C7 B5 A2 C8 B8 A7 D1 C2  
00000AE0:B2 E0 D3 C6 E4 D8 CA E2 D5 C6 71 7B 4B 42 58 10  
00000AF0:42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42  
00000B00:5D 0D 41 5C 0C 41 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00000B10:0C 45 5B 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00000B20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00000B30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00000B40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A  
00000B50:45 5C 0A 45 5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00  
00000B60:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F  
00000B70:5F 0E 3F 5F 0E 40 5D 0D 3F 5F 0E 3F 5F 0E 41 5E  
00000B80:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
00000B90:40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40  
00000BA0:5D 0D 40 5D 0D 41 5E 0E 4B 60 17 A0 AB 83 92 A2  
00000BB0:73 AC B6 94 F4 ED E7 E2 CF BE DA CA BA D7 C4 B2  
00000BC0:D0 86 58 CA 71 3D C9 69 32 C6 5D 24 C4 54 17 CC  
00000BD0:86 5D D5 C5 B2 D2 BC A6 C9 B3 9D C2 AD 96 C1 AC  
00000BE0:97 C8 B1 9A D9 C8 B6 D3 C4 B4 D1 C3 B3 DE D1 C4

00000BF0:E0 D3 C5 E3 D5 C6 A3 A4 81 48 5E 13 42 5D 0D 42  
00000C00:5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 42 5D 0D 41 5C  
00000C10:0C 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C  
00000C20:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00000C30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00000C40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00000C50:43 5C 0C 43 5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45  
00000C60:5C 0A 45 5C 0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F  
00000C70:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F  
00000C80:0E 40 5D 0D 3E 5E 0D 3E 5E 0D 40 5D 0D 40 5D 0D  
00000C90:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00000CA0:5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00000CB0:0D 41 5E 0E 52 64 1F A4 AF 8A 95 A3 75 B3 AC 84  
00000CC0:DC AB 87 D7 A3 7D DB CD BD D2 A1 7F CD 76 42 CB  
00000CD0:71 3C C9 69 32 C7 62 2A C5 5B 20 C3 54 19 D8 AD  
00000CE0:90 E2 D7 CA D3 BE A9 CA B6 A0 D0 BD AA D1 BD A9  
00000CF0:DD CD BD DD D0 C2 D0 C1 B1 DA CD BE DE D1 C3 DB  
00000D00:CD BC 91 9A 6E 4E 62 18 42 5D 0D 42 5D 0D 42 5D  
00000D10:0D 42 5D 0D 42 5E 0B 41 5C 0C 41 5C 0C 41 5C 0C  
00000D20:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0C 43  
00000D30:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00000D40:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00000D50:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00000D60:5C 0C 43 5C 0C 45 5C 0A 45 5C 0A 45 5C 0A 45 5C  
00000D70:0A 45 5C 0A 45 5B 0C 00 3F 5F 0E 3F 5F 0E 3F 5F  
00000D80:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D  
00000D90:3E 5D 0D 3E 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00000DA0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00000DB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0E  
00000DC0:58 67 26 A4 B1 8B B8 B1 8A D6 9F 74 D5 98 6B D7  
00000DD0:A4 7F DD CF BF D2 90 66 CD 77 43 CA 6E 38 C8 65  
00000DE0:2C C6 60 27 C6 5B 21 C3 56 1B CB 71 40 DD C8 B6  
00000DF0:D5 C3 AF E1 D2 C2 DF D0 C1 D9 C8 B7 E1 D3 C4 E4  
00000E00:D8 CA D1 C2 B2 E0 D4 C7 E7 DA CD AD AA 90 6F 7F  
00000E10:47 52 63 1F 41 5B 0D 41 5C 0C 41 5C 0C 42 5D 0D  
00000E20:42 5D 0B 41 5C 0C 41 5C 0C 41 5B 0C 42 5B 0C 43  
00000E30:5B 0C 43 5C 0C 43 5B 0B 43 5C 0C 43 5C 0C 43 5B  
00000E40:0C 43 5B 0C 43 5B 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00000E50:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00000E60:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00000E70:0C 44 5C 0A 44 5C 0A 45 5B 0A 45 5B 0A 45 5B 0A  
00000E80:45 5B 0B 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E  
00000E90:3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40  
00000EA0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00000EB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E  
00000EC0:40 5D 0D 40 5D 0D 40 5D 0D 42 5C 0E 60 6E 33 A9  
00000ED0:B6 91 CD B0 8B D7 A2 78 D6 99 6D D9 A8 86 E2 D7  
00000EE0:CA DE BC A2 CC 78 44 D0 85 57 D2 8B 61 CC 74 42  
00000EF0:CC 76 46 CD 7D 4F CC 76 47 CF 9C 7A D8 C9 B7 DF  
00000F00:D0 C0 DF CF BF DD CE BE E4 D7 C9 EB E0 D4 D1 C3  
00000F10:B3 DC CF C1 F1 E6 DA C8 C0 AE 72 80 4F 56 66 24  
00000F20:42 5B 0E 41 5C 0C 41 5C 0C 42 5D 0D 43 5C 0C 43  
00000F30:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C  
00000F40:0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B 0D 43 5B 0D  
00000F50:43 5B 0D 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00000F60:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00000F70:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00000F80:43 5C 0C 45 5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00

00000F90:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F  
00000FA0:5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00000FB0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
00000FC0:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40  
00000FD0:5D 0D 40 5D 0D 43 5C 10 78 82 4C B9 BF 9C D8 AB  
00000FE0:84 D6 A3 7A D6 A2 7C DE BE A5 E7 DA CD E8 DA CB  
00000FF0:DA AA 89 E7 D9 CC DB AE 91 C7 63 2B C6 5D 24 C4  
00001000:56 1B C2 51 15 D1 A2 83 E1 D4 C5 E1 D2 C2 E1 D1  
00001010:C2 D1 C0 AD D1 C2 B1 E2 D6 C9 D3 C5 B5 D4 C6 B7  
00001020:E2 D6 C8 EA DE D2 95 9A 76 5B 6B 2A 44 5C 0F 41  
00001030:5C 0C 41 5C 0C 41 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00001040:0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B  
00001050:43 5C 0C 43 5C 0C 43 5B 0D 43 5B 0D 43 5B 0D 43  
00001060:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00001070:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00001080:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45  
00001090:5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F  
000010A0:5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F  
000010B0:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
000010C0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
000010D0:5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D  
000010E0:0D 47 5E 13 8E 97 67 C4 BF 9F CE A9 81 CA B7 99  
000010F0:E4 DA CF DB CC BC E9 DA CC E8 DA CC E9 DB CE E5  
00001100:CF BE CB 73 40 C6 5F 26 C6 5C 22 C3 52 15 CA 75  
00001110:45 D9 CA B9 E6 D8 CA E3 D5 C6 D1 BF AE C3 AF 9A  
00001120:CE BE AD D1 C3 B3 CC BD AD D2 C4 B5 D9 CC BD DD  
00001130:CE BF CE C6 B3 6A 78 3A 47 5C 11 41 5C 0C 41 5C  
00001140:0C 42 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D  
00001150:42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43  
00001160:5C 0C 43 5B 0D 43 5B 0D 43 5B 0D 43 5C 0C 43 5C  
00001170:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00001180:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00001190:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 45 5B  
000011A0:0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F  
000011B0:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D  
000011C0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
000011D0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
000011E0:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 4C 61 17  
000011F0:9C A5 7A D1 BF 9E D6 A5 7D E2 BD 9F E3 CF BE DB  
00001200:CB BB E9 DA CC E8 D9 CB EA DF D3 DC B0 93 C7 63  
00001210:29 C5 5C 22 C5 59 1F C5 5B 21 DD BC A4 DE D2 C4  
00001220:E6 D8 CA E8 DA CC CD BB A9 C6 B3 9F CE BD AB CA  
00001230:B8 A5 CF BE AC D5 C4 B2 BD B4 9D AB AC 8B B6 BA  
00001240:9A 78 86 4D 4A 5E 14 43 5C 0C 43 5C 0C 43 5C 0C  
00001250:43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43  
00001260:5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B  
00001270:0D 42 5A 0C 43 5B 0D 43 5C 0C 43 5C 0C 43 5C 0C  
00001280:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00001290:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
000012A0:0C 43 5C 0C 43 5C 0C 45 5B 0C 45 5B 0C 45 5B 0C  
000012B0:45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E  
000012C0:3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40  
000012D0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
000012E0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E  
000012F0:40 5D 0D 40 5D 0D 40 5D 0D 53 64 1D A2 AA 82 D7  
00001300:BC 99 D8 A7 7F DA A8 81 E3 CE BB E1 D2 C3 E8 D9  
00001310:CC E8 D9 CB E4 D6 C7 E6 D7 C9 E2 BC A3 D2 84 57  
00001320:C5 56 1A D4 9A 76 E6 DE D2 E0 D0 C1 E3 D5 C6 F0



00001330:E4 D7 E1 D2 C3 CA B7 A4 D1 C1 B0 C9 B7 A3 D0 BD  
00001340:A9 D8 C6 B2 B5 AC 92 7B 89 53 75 89 4D 70 82 46  
00001350:53 61 1C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00001360:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C  
00001370:0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C  
00001380:42 5A 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00001390:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
000013A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
000013B0:43 5C 0C 45 5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00  
000013C0:3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F  
000013D0:5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
000013E0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
000013F0:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40  
00001400:5D 0D 41 5D 0E 59 67 24 A9 B2 8C D1 BB 98 D7 A5  
00001410:7C D4 B0 8D DE D4 C5 DA CA BB DE D1 C5 DB D0 C3  
00001420:DC CE BF E4 D5 C5 EC E2 D7 F0 E5 D9 E4 C4 AE D7  
00001430:C8 B9 D4 C7 BA D8 C9 BB E0 D2 C4 E9 DD D0 E3 D4  
00001440:C6 CF BD AB D2 C3 B3 C2 B2 A1 C6 B6 A3 CD BA A7  
00001450:CD BD A7 8B 97 65 7D 8E 55 74 85 49 5A 66 21 44  
00001460:5C 0E 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00001470:0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B  
00001480:43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 42 5A 0C 42  
00001490:5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
000014A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
000014B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45  
000014C0:5B 0C 45 5B 0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F  
000014D0:5F 0E 3F 5F 0C 3F 5F 0C 3F 5F 0C 3F 5F 0E 40 5E  
000014E0:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
000014F0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00001500:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 43 5D  
00001510:0F 69 71 33 B3 BC 99 CD BA 96 D6 A4 79 A2 9B 72  
00001520:B3 AE 91 DE D0 C2 D9 C2 AF CE 9D 7C CF 8E 64 CE  
00001530:7D 4C D6 AB 8D DD D4 C9 DB D1 C6 DB CF C2 D8 CA  
00001540:BC CF C1 B2 D1 C4 B6 DB CF C3 D6 C8 BB C6 B6 A5  
00001550:CF C2 B4 D4 C8 BC A1 9B 81 69 6C 3C 60 63 2E 6C  
00001560:77 41 7A 88 54 74 83 4B 60 6C 27 45 5C 0E 43 5C  
00001570:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D  
00001580:42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43  
00001590:5C 0C 43 5B 0D 44 5A 0C 44 5A 0C 44 5B 0B 43 5C  
000015A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
000015B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
000015C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0C 43 5B  
000015D0:0C 45 5B 0C 43 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60  
000015E0:0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D  
000015F0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00001600:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00001610:0D 40 5D 0D 40 5D 0D 40 5D 0D 45 5E 10 7D 82 48  
00001620:BE C5 A4 B9 B0 8A BA B2 93 D1 C9 B6 DB AC 89 E6  
00001630:CC B7 E8 D2 C0 D1 85 55 CC 75 41 CA 6C 36 C6 6E  
00001640:3B B5 A4 8E D2 C6 B8 E7 D8 C9 E5 D6 C7 E0 D1 C2  
00001650:DA CC BD DB CD C0 D3 C6 B7 BE AF A0 D0 C3 B5 E9  
00001660:DE D2 EA E2 D6 C3 BE A9 8B 89 64 61 66 34 65 6D  
00001670:3E 68 76 45 63 6C 2D 47 5D 0F 43 5C 0C 43 5C 0C  
00001680:43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43  
00001690:5B 0D 43 5C 0C 42 5B 0B 43 5C 0C 43 5C 0C 43 5B  
000016A0:0D 45 5B 0D 45 5B 0C 45 5B 0C 43 5C 0C 42 5B 0B  
000016B0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
000016C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C

000016D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C  
000016E0:43 5D 0A 00 3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C  
000016F0:3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40  
00001700:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00001710:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
00001720:40 5D 0D 40 5D 0D 47 5E 12 91 93 61 BC C2 A1 C3  
00001730:C4 AB E8 E0 D4 DF B4 93 D4 96 68 DE B2 93 E0 BB  
00001740:A0 CF 81 50 CD 7A 48 CB 71 3C C8 63 2A D3 9A 75  
00001750:E1 D5 C7 DB CB BB D8 C7 B7 D7 C6 B5 D8 C7 B6 D8  
00001760:C7 B6 D3 C1 AF C5 B2 9E DB CD BE E9 DE D3 EF E5  
00001770:DB F6 EE E4 FA F0 E7 E3 D8 CA BB B0 9A A1 9D 83  
00001780:71 74 3B 4A 5E 12 43 5C 0C 43 5C 0C 43 5C 0C 43  
00001790:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C  
000017A0:0C 42 5B 0B 43 5B 0C 43 5B 0C 43 5C 0C 45 5B 0C  
000017B0:44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 43 5C 0C 43  
000017C0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
000017D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
000017E0:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 43 5C 0A 00  
000017F0:3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C 3F 60 0C 3F  
00001800:5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00001810:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
00001820:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00001830:5D 0D 4E 60 17 9B 9C 6F BB C2 9E D7 D5 C2 E0 C0  
00001840:A6 D7 A0 76 D6 9C 72 D8 A1 7A D4 95 6A D0 82 51  
00001850:CD 7A 47 CA 6D 38 D7 A7 88 DB BF AA D9 C9 B8 D3  
00001860:C3 B2 D0 BF AC D0 BE AC D1 C0 AE CF BD AA C6 B3  
00001870:9F D1 BF AD E2 D5 C7 E6 DC D0 EA DF D4 E9 DE D3  
00001880:E4 D8 CC DC CE C0 D1 C0 AF CC BD AA 73 79 3E 50  
00001890:5F 17 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
000018A0:0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B  
000018B0:43 5B 0D 43 5B 0D 43 5C 0C 45 5B 0C 44 5A 0B 44  
000018C0:5A 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C  
000018D0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
000018E0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
000018F0:5C 0C 43 5C 0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F  
00001900:5F 0E 3F 60 0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E  
00001910:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
00001920:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00001930:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 57 64  
00001940:1E A5 A5 7C B8 BE 9B D4 D1 BF E2 C5 AD D8 A2 78  
00001950:D7 9E 74 D8 A4 7E D2 8C 5E D0 83 53 CC 74 40 D1  
00001960:8D 62 DE D0 C2 DC CE C0 D4 C2 B1 CF BD AB D1 BF  
00001970:AD D4 C3 B2 D4 C3 B1 CF BE AB D1 BF AD DC CC BC  
00001980:DE D0 C3 DF D3 C6 DD D0 C3 DC CF C1 DF D3 C5 DA  
00001990:CC BF D3 C5 B6 90 8D 58 60 6D 27 56 64 1E 43 5C  
000019A0:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D  
000019B0:42 5A 0C 43 5B 0D 43 5C 0C 42 5B 0B 43 5B 0D 43  
000019C0:5B 0D 43 5C 0C 44 5A 0B 44 5A 0B 44 5A 0B 42 5B  
000019D0:0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
000019E0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
000019F0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00001A00:0C 45 5B 0C 45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60  
00001A10:0C 3F 60 0C 3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D  
00001A20:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00001A30:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00001A40:0D 40 5D 0D 40 5D 0D 40 5D 0D 62 6B 26 B0 B1 8C  
00001A50:BB BE A0 E3 DD D0 E8 D7 C8 D8 A2 7A D7 A4 7D D5  
00001A60:9C 74 D1 87 57 CE 7C 4A D0 88 5C D7 C2 AF D4 C5

00001A70:B5 D5 C4 B2 D9 C9 B8 DD CD BD DC CC BC DA C9 B9  
00001A80:D7 C6 B6 D7 C6 B5 DF D0 C0 D7 C7 B6 D7 C8 B9 D5  
00001A90:C7 B9 D9 CB BD E5 DA CD E4 D9 CC E0 D3 C6 D7 CD  
00001AA0:BD 7C 84 48 5B 6A 1F 5C 67 21 43 5C 0C 43 5C 0C  
00001AB0:43 5C 0C 43 5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43  
00001AC0:5B 0D 43 5C 0C 42 5B 0B 43 5B 0D 42 5A 0C 42 5B  
00001AD0:0B 44 5A 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B  
00001AE0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00001AF0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00001B00:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C  
00001B10:45 5C 0A 00 3F 5F 0E 3F 5F 0E 3F 60 0C 3F 60 0C  
00001B20:3F 60 0C 3F 5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40  
00001B30:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00001B40:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
00001B50:40 5D 0D 41 5D 0D 6D 73 32 B9 BA 9B D6 D1 BF EC  
00001B60:E3 DA EE E7 DF E1 C0 A5 D7 AB 88 D7 A3 7E DA A7  
00001B70:84 DD B1 94 D9 C4 B1 D4 C5 B4 DB CB BB E9 DB CD  
00001B80:E9 DB CE E3 D5 C6 DE CF BF DC CD BC DF CF CE E1  
00001B90:D1 C2 DF CF C0 DA CA B9 D6 C7 B8 D5 C7 B9 E8 DC  
00001BA0:D1 E8 DC D1 E4 D8 CC E5 D9 CC B2 AD 93 7D 89 53  
00001BB0:6A 77 32 60 6A 24 44 5C 0C 43 5C 0C 43 5C 0C 43  
00001BC0:5C 0C 43 5C 0C 43 5B 0D 42 5A 0C 43 5B 0D 43 5C  
00001BD0:0C 42 5B 0B 42 5A 0C 42 5A 0C 42 5B 0B 44 5A 0B  
00001BE0:44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 43 5C 0C 43  
00001BF0:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00001C00:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00001C10:43 5C 0C 43 5C 0C 43 5C 0C 45 5B 0C 45 5C 0A 00  
00001C20:40 5D 0D 41 5E 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F  
00001C30:5F 0E 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00001C40:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
00001C50:41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41  
00001C60:5E 0F 77 79 3C CC CA B4 EB E2 D9 ED E4 DB EF E7  
00001C70:E0 EF E7 DE DE CC BB DD CD BC E8 DB CD E9 DC CF  
00001C80:DC CD BD E5 D6 C7 F1 E4 D8 F0 E3 D6 EA DC CE E5  
00001C90:D6 C7 E1 D2 C2 DF CF C0 DF CF C0 DD CD BD D9 C8  
00001CA0:B8 D7 C6 B5 E0 D3 C6 E6 DB CF EB E0 D5 E1 D5 C8  
00001CB0:DF D3 C7 E4 D8 CB A7 A3 88 77 82 4F 77 86 4B 6B  
00001CC0:71 30 44 5D 0E 41 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00001CD0:0C 43 5C 0C 43 5C 0C 43 5B 0D 43 5B 0D 43 5B 0D  
00001CE0:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A 0B 44  
00001CF0:5A 0B 42 5B 0B 42 5B 0B 44 5A 0B 43 5C 0C 43 5C  
00001D00:0C 43 5D 0A 43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C  
00001D10:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43  
00001D20:5C 0C 43 5C 0C 42 5B 0B 44 5B 09 00 40 5D 0D 41  
00001D30:5E 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 41 5E  
00001D40:0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E  
00001D50:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00001D60:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 47 5F 13 80 7F  
00001D70:46 DE DA C9 EF E6 DD EE E5 DD F0 E8 E0 F1 E8 E0  
00001D80:E2 D2 C3 DE CD BD E3 D3 C3 E3 D4 C5 E2 D3 C5 E6  
00001D90:D9 CA E5 D7 C9 E7 DA CB E9 DA CC E5 D6 C8 E2 D3  
00001DA0:C4 E0 D1 C2 DA CA B9 D3 C2 B1 CF BD AA CC B9 A6  
00001DB0:D6 C7 B8 E1 D5 C9 E7 DB CF DF D3 C6 E0 D3 C7 DE  
00001DC0:D1 C5 D4 C9 B9 7F 85 5B 75 83 4A 74 78 3A 4A 5F  
00001DD0:14 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00001DE0:42 5B 0B 42 5B 0C 42 5B 0C 43 5B 0C 42 5B 0B 42  
00001DF0:5B 0B 42 5B 0B 42 5B 0B 43 5A 0B 43 5A 0B 42 5B  
00001E00:0B 42 5B 0B 44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A

00001E10:43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43  
00001E20:5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00001E30:0C 42 5B 0B 44 5B 09 00 40 5D 0D 41 5E 0E 3F 5F  
00001E40:0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 41 5E 0E 40 5D 0D  
00001E50:40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40  
00001E60:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00001E70:0D 40 5D 0D 40 5D 0D 4B 61 15 8E 8B 57 CF CE B6  
00001E80:EA E2 D6 F0 E7 DF F1 E9 E1 F1 E8 E0 E4 D5 C5 E1  
00001E90:D1 C0 E0 D0 C0 E0 D0 C2 E4 D5 C6 E3 D4 C5 E3 D4  
00001EA0:C5 E5 D6 C8 E8 DB CD EB DD D0 EB DD CF E8 DA CC  
00001EB0:E0 D1 C2 D7 C6 B5 CE BB A9 C9 B5 A2 D3 C4 B5 D3  
00001EC0:C4 B6 DA CD BF E1 D5 C9 E3 D7 CB DD D1 C4 EA DF  
00001ED0:D4 B7 B4 9C 77 85 4D 76 7D 3E 4F 61 17 41 5C 0C  
00001EE0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 42  
00001EF0:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B  
00001F00:0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5A 0B 43 5A 0B  
00001F10:44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A 43 5D 0A 43  
00001F20:5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00001F30:0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 43 5A 0B  
00001F40:44 5B 09 00 40 5D 0D 40 5D 0D 3F 5F 0E 3F 5F 0E  
00001F50:3F 5F 0E 3F 5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40  
00001F60:5D 0D 40 5D 0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D  
00001F70:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
00001F80:40 5D 0D 50 63 1A A3 9F 71 CA C9 B1 E0 D9 CB F1  
00001F90:E8 E0 F2 EA E2 F2 E9 E0 E7 D8 C8 E4 D4 C4 E1 D0  
00001FA0:C1 E4 D5 C6 DD CD BD D7 C6 B6 DE CF BF E5 D6 C8  
00001FB0:EB DE D0 F0 E3 D6 F1 E5 D8 F0 E3 D6 ED DF D3 E9  
00001FC0:DB CD E8 DA CC E3 D4 C5 E3 D7 CA E4 D8 CC D9 CB  
00001FD0:BD D9 CB BD E0 D3 C7 DF D3 C6 E7 DC D0 E1 D8 C8  
00001FE0:80 8D 56 7A 82 45 55 64 1B 41 5C 0C 43 5C 0C 43  
00001FF0:5C 0C 43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B  
00002000:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B  
00002010:42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B 44 5A 0B 43  
00002020:5C 0C 43 5C 0C 43 5D 0A 43 5D 0A 43 5D 0A 43 5C  
00002030:0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00002040:43 5C 0C 43 5C 0C 43 5A 0B 44 5A 0B 44 5B 09 00  
00002050:40 5D 0D 40 5D 0D 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F  
00002060:5F 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
00002070:0D 41 5E 0E 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
00002080:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 58  
00002090:67 20 B2 AF 88 D8 D4 C5 F0 E7 DE F1 E9 E1 F3 EC  
000020A0:E4 F3 E9 E1 E9 DA CB E6 D6 C7 E4 D4 C4 E6 D8 C9  
000020B0:DF D0 C0 D5 C4 B3 DB CB BA E5 D7 C9 EB DD D0 EF  
000020C0:E2 D5 F1 E4 D8 F0 E4 D7 ED DF D2 EE E1 D4 E6 D8  
000020D0:C9 C1 AE 98 CA B9 A8 E8 DC D1 EC E1 D7 E7 DC D0  
000020E0:DD D0 C3 D4 C6 B7 D8 CB BC AE AC 8B 79 87 4D 7B  
000020F0:84 4A 5C 67 20 41 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00002100:0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B  
00002110:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42  
00002120:5B 0B 44 5A 0B 44 5A 0B 44 5A 0B 43 5C 0C 43 5C  
00002130:0C 43 5D 0A 43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C  
00002140:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42  
00002150:5B 0B 44 5A 0B 44 5A 0B 44 5B 09 00 40 5D 0D 40  
00002160:5D 0D 3F 5F 0E 3F 5F 0E 3F 5F 0E 3F 5F 0E 40 5D  
00002170:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 41 5E 0E  
00002180:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00002190:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 64 6C 29 C3 BD  
000021A0:A0 F4 EC E3 F2 E9 E0 F3 EA E2 F5 EE E6 F3 EA E1

000021B0:EB DC CD E8 D9 CA E5 D5 C6 E4 D4 C5 E3 D3 C3 D9  
000021C0:C8 B8 D3 C1 AF DC CC BC EB DD D0 EF E2 D6 F2 E5  
000021D0:DA F2 E5 D9 F4 E7 DB EA DC CE C7 B3 A0 B7 A2 8B  
000021E0:D1 C2 B3 EB E0 D6 E6 DA CF EA E0 D5 EF E4 DB E8  
000021F0:DC D1 D0 C5 B4 8A 8F 63 7C 8B 51 79 85 4A 64 69  
00002200:27 41 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 42 5B 0B  
00002210:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42  
00002220:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A  
00002230:0B 44 5A 0B 44 5A 0B 43 5C 0C 43 5C 0C 43 5D 0A  
00002240:43 5D 0A 43 5D 0A 43 5C 0C 43 5C 0C 43 5C 0C 43  
00002250:5C 0C 43 5C 0C 43 5C 0C 42 5B 0B 44 5A 0B 44 5A  
00002260:0B 44 5A 0B 44 5B 09 00 40 5D 0D 40 5D 0D 3F 5E  
00002270:0D 3F 5E 0D 3F 5E 0D 3F 5E 0D 40 5D 0D 40 5D 0D  
00002280:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00002290:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
000022A0:0D 3F 5C 0C 40 5C 0D 72 74 35 CC C5 AC F4 EC E4  
000022B0:F1 E9 E0 F6 EE E6 F5 EE E6 F2 E9 DE ED DE CF EB  
000022C0:DC CE E3 D3 C3 D8 C7 B4 DB CB BA DF D0 BF D9 C8  
000022D0:B7 D9 C8 B7 ED DE D0 F4 E6 D9 F9 ED E0 FD F0 E4  
000022E0:FF F3 E7 E0 CF BD CA B6 A1 C7 B3 9D F3 E7 DC F4  
000022F0:EA E0 E7 DD D0 E7 DC D0 E8 DD D2 EF E5 DB F0 E7  
00002300:DD A5 A6 81 7C 8A 50 76 83 47 6F 71 31 43 5C 0E  
00002310:43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42  
00002320:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B  
00002330:0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B  
00002340:43 5A 0B 42 5B 0B 42 5B 0B 43 5C 0A 43 5C 0A 43  
00002350:5C 0A 43 5C 0C 43 5C 0C 43 5C 0B 43 5C 0B 43 5C  
00002360:0C 43 5B 0C 42 5A 0A 43 59 0A 44 5A 0B 44 5A 0B  
00002370:43 5B 09 00 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
00002380:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00002390:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
000023A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C  
000023B0:43 5E 0F 80 7E 41 D3 CD BA F2 E9 E0 F4 EB E2 F4  
000023C0:EA E1 F0 E8 DF E5 D9 CD DD CE BE E4 D5 C7 DC CD  
000023D0:BF D9 CA BB D9 CA BB DA CC BD D2 C4 B6 CC BF B2  
000023E0:CC C1 B5 C8 BD B2 C3 B9 AF C0 B6 AB BA AE A2 AA  
000023F0:9A 8B A6 95 84 BB AD 9E D7 CE C5 D5 CA BF D3 C6  
00002400:B9 DC CF C1 E5 D7 C9 EB DF D2 E6 DC CF 8D 92 67  
00002410:7A 87 4D 73 82 45 79 79 39 49 5E 11 43 5C 0C 42  
00002420:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B  
00002430:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B  
00002440:42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42  
00002450:5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00002460:0C 43 5C 0C 43 5D 0A 43 5D 0A 43 5C 0C 44 5A 0B  
00002470:43 59 0A 43 59 0A 44 5A 0B 44 5A 0B 42 5C 09 00  
00002480:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
00002490:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
000024A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
000024B0:40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C 45 5F 11 90  
000024C0:8A 4F EA E1 D5 E8 DF D7 E7 DF D6 E7 E1 DA E5 E0  
000024D0:DB E6 E3 DF EA E8 E5 F0 EE ED F2 F2 F1 F1 F1 F1  
000024E0:F0 F0 F0 EB EB EB E3 E4 E4 D7 D7 D8 C2 C3 C4 A8  
000024F0:A9 AA 8B 8B 8C 6C 6C 6D 53 54 55 45 45 46 3E 3F  
00002500:3F 3B 3B 3B 3C 3B 3A 41 40 3F 4A 48 46 5B 57 54  
00002510:78 72 6C 9C 94 8B B4 AB A2 A2 9F 86 95 9C 6F 7A  
00002520:8A 4F 7B 7D 3D 4B 5F 13 43 5C 0C 42 5B 0B 42 5B  
00002530:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B  
00002540:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42

00002550:5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B  
00002560:0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
00002570:43 5D 0A 43 5D 0A 43 5C 0C 44 5A 0B 43 59 0A 43  
00002580:59 0A 44 5A 0B 44 5A 0B 42 5C 09 00 40 5D 0D 40  
00002590:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
000025A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
000025B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
000025C0:5D 0D 40 5D 0D 3F 5C 0C 52 64 1D A1 92 80 C5 C5  
000025D0:C4 D2 D2 D2 E1 E2 E3 EB EC ED F2 F2 F3 F5 F5 F5  
000025E0:F7 F7 F6 F6 F6 F6 F5 F4 F3 F4 F3 F1 F1 EF EE F1  
000025F0:EF ED ED EC EA E7 E5 E3 DF DE DC D2 D1 D0 C4 C2  
00002600:C0 B7 B5 B2 AB A9 A6 A0 9D 9B 93 90 8D 89 88 87  
00002610:7E 7E 7F 72 72 71 6B 6A 6A 69 69 6A 69 6A 6B 6F  
00002620:71 73 83 86 89 A2 A3 A5 BB BA BA C4 C7 BF AE A4  
00002630:81 55 64 1C 43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B  
00002640:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42  
00002650:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A  
00002660:0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C  
00002670:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5D 0A 43  
00002680:5D 0A 43 5C 0C 44 5A 0B 44 5A 0B 43 59 0A 43 59  
00002690:0A 44 5A 0B 42 5C 09 00 40 5D 0D 40 5D 0D 40 5D  
000026A0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
000026B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
000026C0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
000026D0:0D 3F 5C 0C 75 82 5E 9E A0 A2 C6 C7 C8 DE DE DE  
000026E0:EA E9 E8 F2 F1 F0 F5 F3 F0 F5 F3 F0 F1 ED E9 EB  
000026F0:E6 E0 E9 E3 DD E6 E1 DA E0 DB D6 E9 E5 DD F6 F4  
00002700:F3 E9 E5 E1 E4 DF DA DC D7 D0 D2 CA C1 D4 CC C4  
00002710:DB D4 CC E2 DC D5 D4 CC C3 CF C7 BC F3 EF EB DF  
00002720:D9 D2 D9 D2 CB DC D7 D1 DB D8 D4 C9 C4 B8 B8 AF  
00002730:96 C3 BF B7 D0 D0 D0 D0 D1 D2 CF D1 D3 86 8F 6D  
00002740:43 5C 0C 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42  
00002750:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B  
00002760:0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B  
00002770:42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43  
00002780:5C 0C 43 5C 0C 43 5C 0C 43 5D 0A 43 5D 0A 44 5A  
00002790:0B 44 5A 0B 44 5A 0B 43 59 0A 43 59 0A 44 5A 0B  
000027A0:42 5C 09 00 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
000027B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
000027C0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
000027D0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C  
000027E0:89 91 7B AD AC AC AD A4 84 A2 97 5E A7 9B 5F EE  
000027F0:E9 DA FC FC FC F8 F6 F4 EE E9 E4 E5 DE D7 E6 E1  
00002800:DB E4 E0 DB B8 AA 7D B7 AD 7C FF FF FF F2 F2 F2  
00002810:E5 E1 DD DE D9 D3 D6 CF C8 DB D6 D0 DB D6 D0 D9  
00002820:D3 CD D1 CA C2 BE B4 A8 BA AF A3 C6 BD B3 CE C6  
00002830:BE D1 CD C7 C5 B8 99 9A 8F 4D 91 88 43 9A 90 4E  
00002840:A5 99 61 B2 A9 8B CB CB CC 9D A4 90 43 5C 0C 42  
00002850:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B  
00002860:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B  
00002870:42 5B 0B 42 5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42  
00002880:5B 0B 42 5B 0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C  
00002890:0C 43 5C 0C 43 5D 0A 43 5D 0A 44 5A 0B 44 5A 0B  
000028A0:43 59 0A 43 59 0A 43 59 0A 44 5A 0B 42 5C 09 00  
000028B0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
000028C0:5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D  
000028D0:0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
000028E0:40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C 90 99 7F 7D

000028F0:7C 7C 73 70 69 83 7D 64 93 88 5F A6 9C 78 EA E8  
00002900:E4 FB F9 F7 E7 E1 DA E1 D9 D1 D9 D0 BF AB 9F 68  
00002910:8D 84 3E 99 8F 4C C8 BB 90 E6 E0 CA F3 F0 EB E9  
00002920:E5 E1 E0 DA D4 E6 E2 DE DD D7 D2 CE C7 BF DF DA  
00002930:D5 E1 DC D7 D5 CE C6 D6 CF C7 CC C3 B4 BC B1 99  
00002940:96 8A 54 8C 80 47 90 84 56 8E 83 5D 89 82 6C 9A  
00002950:97 91 B2 B2 B3 92 9B 80 43 5C 0C 42 5B 0B 42 5B  
00002960:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B  
00002970:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42  
00002980:5B 0B 44 5A 0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B  
00002990:0B 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C  
000029A0:43 5D 0A 42 5C 09 44 5A 0B 44 5A 0B 43 59 0A 43  
000029B0:59 0A 44 5A 0B 44 5A 0B 42 5C 09 00 40 5D 0B 40  
000029C0:5D 0B 40 5E 0C 40 5E 0C 40 5E 0C 40 5E 0C 40 5E  
000029D0:0C 40 5D 0B 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D  
000029E0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40  
000029F0:5D 0D 40 5D 0D 3F 5C 0C 49 63 1A 85 94 6B A3 A7  
00002A00:9E 9F A1 A3 9E A1 A5 A3 A4 A6 B9 B8 B7 EA E7 E5  
00002A10:DF D6 CD CE C8 C2 B6 B0 A3 A8 9F 81 A7 9F 81 A2  
00002A20:98 77 9A 8F 66 9A 91 6B A2 9B 84 B5 B4 AB D4 D1  
00002A30:CD DE D9 D3 DE D8 D1 D0 C9 C1 D0 CA C4 DF DD DC  
00002A40:E4 E1 DE CB C8 C6 7E 7A 71 57 52 49 56 53 50 5B  
00002A50:5C 5E 6E 71 74 86 88 8B A0 A1 A3 A6 AB A0 82 8F  
00002A60:66 4C 63 18 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B  
00002A70:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42  
00002A80:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 44 5A  
00002A90:0B 44 5A 0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C  
00002AA0:43 5C 0C 43 5C 0C 43 5C 0C 43 5C 0C 43 5B 09 43  
00002AB0:5B 09 44 5A 09 44 5A 0B 43 59 0A 43 59 0A 43 59  
00002AC0:0A 43 59 0A 43 5A 0A 00 40 5D 0D 40 5E 0E 41 5E  
00002AD0:0E 41 5E 0E 41 5E 0E 41 5E 0E 40 5D 0D 40 5D 0D  
00002AE0:40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 40 5D 0D 3F  
00002AF0:5C 0C 3F 5C 0C 3F 5C 0C 3F 5C 0C 40 5D 0D 40 5D  
00002B00:0D 40 5D 0D 3F 5C 0C 3F 5C 0C 44 60 13 58 70 2E  
00002B10:73 85 53 84 92 6B 8F 99 7C A6 AC 9A B7 B8 AE B9  
00002B20:B9 B8 BB BD BE BA BD BF B8 BB BD B5 B8 BB B0 B3  
00002B30:B6 A9 AB AD 9D 9E 9F 8F 8F 90 86 86 86 87 86 86  
00002B40:B5 B1 AD DE D8 D1 BB B5 AF 6C 6C 79 79 79 6F  
00002B50:6F 6F 71 72 71 7A 7C 76 7D 83 73 7C 86 68 7B 86  
00002B60:60 6F 7E 4D 57 6B 2A 46 5E 11 43 5C 0C 43 5D 0A  
00002B70:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42  
00002B80:5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5B  
00002B90:0B 42 5B 0B 42 5B 0B 42 5B 0B 42 5A 0B 42 5A 0B  
00002BA0:42 5B 0B 42 5B 0B 42 5B 0B 42 5B 0B 43 5C 0C 43  
00002BB0:5C 0A 43 5C 0A 43 5C 0A 44 5B 09 44 5A 0A 44 5A  
00002BC0:0A 44 5A 0B 44 5A 0B 43 59 0A 43 59 0A 43 59 0A  
00002BD0:44 5A 0B 00 3E 5E 0D 3F 5D 0D 3F 5D 0D 40 5E 0E  
00002BE0:41 5E 0E 40 5D 0D 40 5D 0D 3F 5C 0C 40 5D 0C 40  
00002BF0:5D 0C 40 5D 0C 40 5D 0C 3F 5C 0B 3F 5C 0B 3F 5C  
00002C00:0B 3F 5C 0B 40 5D 0D 40 5D 0D 40 5D 0D 3F 5C 0C  
00002C10:3F 5C 0C 3F 5C 0C 40 5D 0D 40 5D 0D 40 5D 0D 40  
00002C20:5D 0D 40 5D 0D 41 5C 0C 48 62 16 52 69 23 58 6E  
00002C30:2D 5D 72 34 61 75 3A 64 77 3F 67 79 42 68 7A 44  
00002C40:69 7B 46 69 7A 46 68 78 44 66 77 42 64 75 40 8C  
00002C50:96 6D C0 BE B0 51 62 2B 57 6B 2D 58 6D 2C 51 67  
00002C60:22 49 60 15 41 5B 0B 41 5B 0B 41 5B 0B 41 5B 0B  
00002C70:41 5B 0B 40 5B 0B 40 5B 0B 40 5C 09 42 5B 0A 42  
00002C80:5B 0A 42 5B 0A 42 5B 0A 42 5B 0A 42 5B 0A 42 5B

00002C90:0A 41 5A 0A 42 5A 0C 42 5A 0C 42 5A 0C 42 5B 0B  
00002CA0:42 5B 0B 42 5B 0B 42 5B 0A 42 5B 0A 44 5A 0B 44  
00002CB0:5A 0B 44 5A 0A 44 5A 0A 44 5A 0A 45 5B 0B 45 5B  
00002CC0:0B 45 5B 0B 44 5A 09 44 5A 09 44 5A 0B 44 5A 0B  
00002CD0:44 5A 0B 44 5A 0B 43 59 0A 43 59 0A 43 5B 09 00  
00002CE0:3F 5F 0D 3F 5F 0D 3F 5F 0D 3F 5F 0D 40 5E 0D 40  
00002CF0:5E 0D 40 5E 0D 40 5E 0D 40 5F 0C 40 5F 0C 40 5F  
00002D00:0C 40 5F 0C 3F 5E 0B 3F 5E 0B 40 5E 0B 40 5E 0B  
00002D10:40 5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40  
00002D20:5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40 5D 0C 40 5D  
00002D30:0C 41 5D 0C 41 5C 0B 41 5C 0B 41 5C 0B 41 5C 0B  
00002D40:41 5D 0C 41 5D 0C 41 5D 0C 40 5C 0B 40 5C 0B 40  
00002D50:5C 0B 40 5C 0B 40 5C 0C 41 5C 0C 41 5C 0C 5A 70  
00002D60:2B 41 5C 0C 41 5C 0C 41 5C 0C 41 5C 0C 41 5C 0C  
00002D70:41 5C 0B 41 5C 0B 41 5C 0B 41 5C 0B 41 5C 0B 41  
00002D80:5C 0B 41 5C 0B 41 5D 0B 42 5D 0A 42 5D 0A 42 5D  
00002D90:0A 42 5D 0A 42 5D 0A 42 5D 0A 42 5D 0A 42 5D 0A  
00002DA0:42 5C 0B 42 5C 0B 42 5C 0B 42 5C 0B 42 5C 0B 43  
00002DB0:5C 0B 43 5D 0A 43 5D 0A 44 5C 0B 44 5C 0B 44 5C  
00002DC0:0A 44 5C 0A 44 5C 0A 44 5C 0A 44 5C 0A 44 5C 0A  
00002DD0:43 5B 0A 43 5B 0A 43 5B 0A 43 5B 0A 43 5B 0A 43  
00002DE0:5B 0A 43 5B 0A 43 5B 0A 43 5D 09 00 3F 61 0C 40  
00002DF0:62 0D 40 62 0D 3F 61 0C 3F 61 0C 3F 61 0C 3F 61  
00002E00:0C 3F 61 0C 3F 61 0C 3F 61 0C 3F 61 0C 3F 61 0C  
00002E10:3F 60 0C 3F 60 0C 40 60 0C 40 60 0C 40 60 0C 40  
00002E20:60 0C 3F 5F 0C 3F 5F 0C 3F 5F 0C 40 60 0C 40 60  
00002E30:0C 40 60 0C 3F 5F 0B 3F 5F 0B 3F 5F 0C 41 5F 0C  
00002E40:41 5E 0B 41 5E 0B 42 5F 0C 42 5F 0C 40 60 0C 40  
00002E50:60 0C 40 60 0C 3F 5F 0B 3F 5F 0B 3F 5F 0B 3F 5F  
00002E60:0B 3F 5F 0B 42 5F 0C 42 5F 0C 41 5E 0B 41 5E 0B  
00002E70:41 5E 0B 41 5E 0B 41 5E 0B 41 5F 0C 41 5E 0B 41  
00002E80:5E 0B 41 5E 0B 41 5E 0B 41 5E 0B 41 5F 0B 41 5F  
00002E90:0B 41 5F 0A 41 5F 0B 41 5F 0B 41 5F 0B 41 5F 0B  
00002EA0:41 5F 0B 41 5F 0B 41 5F 0B 41 5F 0B 42 5F 0B 42  
00002EB0:5F 0B 41 5F 0B 41 5F 0B 41 5F 0B 43 5F 0B 43 5E  
00002EC0:0A 43 5E 0A 43 5F 0B 43 5F 0B 43 5F 0B 43 5F 0B  
00002ED0:43 5F 0B 43 5F 0A 43 5F 0A 43 5F 0A 42 5E 09 42  
00002EE0:5E 09 42 5E 09 42 5E 09 42 5E 09 42 5E 09 42 5E  
00002EF0:09 42 5E 09 42 5E 09 00 3F 64 0B 3F 64 0B 3F 64  
00002F00:0B 3F 64 0B 3F 64 0B 3F 64 0B 3F 64 0B 3F 64 0B  
00002F10:3F 64 0C 3F 64 0C 3F 64 0C 3F 62 0C 40 62 0C 40  
00002F20:62 0C 40 62 0C 40 62 0C 40 62 0C 40 62 0C 3F 62  
00002F30:0C 3F 62 0C 40 63 0C 40 63 0D 40 62 0C 40 62 0C  
00002F40:3F 62 0B 3F 62 0B 3F 62 0C 40 62 0C 40 62 0C 41  
00002F50:62 0C 41 63 0D 40 62 0C 40 63 0A 40 63 0A 40 63  
00002F60:0A 3F 62 0C 3F 62 0C 3F 62 0C 3F 62 0C 3F 62 0C  
00002F70:40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40  
00002F80:61 0B 40 61 0B 41 62 0C 40 61 0B 40 61 0B 40 61  
00002F90:0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B  
00002FA0:40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40  
00002FB0:61 0B 40 61 0B 40 61 0B 40 61 0B 40 61 0B 40 61  
00002FC0:0B 40 62 09 40 62 09 40 62 09 42 61 09 42 61 09  
00002FD0:42 61 09 42 61 09 42 61 09 42 61 09 42 61 09 42  
00002FE0:61 09 42 61 09 42 61 09 40 62 09 42 61 09 42 61  
00002FF0:09 42 61 09 42 61 09 42 61 09 42 61 09 42 61 09  
00003000:42 61 09 00 3D 60 0F 3D 60 0F 3D 60 0F 3D 60 0F  
00003010:3D 60 0F 3D 60 0F 3D 60 0F 3D 60 0F 3D 60 0F 3D  
00003020:60 0F 3D 60 0F 3D 60 0F 3D 60 0F 3D 60 0F 3D 60



```

00003030:0F 3D 60 0F 3D 61 0E 3D 61 0E 3C 60 0D 3C 60 0D
00003040:3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D
00003050:61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61
00003060:0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E 3D 61 0E
00003070:3C 60 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E
00003080:5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F
00003090:0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D
000030A0:3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E
000030B0:5F 0D 3E 5F 0D 3E 5F 0D 3F 5F 0D 3F 5F 0D 3F 5F
000030C0:0D 3F 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D 3E 5F 0D
000030D0:3E 60 0B 3D 5F 0A 3E 5F 0A 3E 5F 0A 3F 60 0B 3F
000030E0:60 0B 3F 60 0B 3F 60 0B 3F 60 0B 3F 60 0B 3F 60
000030F0:0B 3F 60 0B 3E 60 0B 3E 60 0B 3E 60 0B 3F 60 0B
00003100:3F 60 0B 3F 60 0B 3F 5F 0D 3F 5F 0D 3D 5F 0A 00

```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x0000004C)																															
Size (0x00002FA8)																															
Bounds (0x00000000)																															
... (0x0000002D)																															
... (0x00000059)																															
... (0x00000059)																															
xDest (0x00000000)																															
yDest (0x0000002D)																															
cxDest (0x0000005A)																															
cyDest (0x0000002D)																															

**Figure 13: EMR\_BITBLT Record Example, Part 1**

**Type:** 0x0000004C identifies this record type as EMR\_BITBLT. This value **MUST** be 0x0000004C.

**Size:** 0x00002FA8 specifies the size of this record in bytes.

**Bounds:** 0x00000000, 0x0000002D, 0x00000059, 0x00000059 specifies the bounding rectangle in device units.

**xDest:** 0x00000000 specifies the logical x-coordinate of the upper-left corner of the destination rectangle.

**yDest:** 0x0000002D specifies the logical y-coordinate of the upper-left corner of the destination rectangle.

**cxDest:** 0x0000005A specifies the logical width of the destination rectangle.

**cyDest:** 0x0000002D specifies the logical height of the destination rectangle.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
BitBltRasterOperation (0x00CC0020)																																	
xSrc (0x00000000)																																	
ySrc (0x00000000)																																	
xformSrc (0x3F800000)																																	
... (0x00000000)																																	
... (0x00000000)																																	
... (0x3F800000)																																	
... (0x00000000)																																	
... (0x00000000)																																	

**Figure 14: EMR\_BITBLT Record Example, Part 2**

**BitBltRasterOperation:** 0x00CC0020 specifies the raster-operation code. These codes define how the color data of the source rectangle is to be combined with the color data of the destination rectangle to achieve the final color. The value **MUST** be in the Windows Metafile Format **Ternary Raster Operation** enumeration, specified in [\[MS-WMF\]](#) section **2.1.32**.

**xSrc:** 0x00000000 specifies the logical x-coordinate of the upper-left corner of the source rectangle.

**ySrc:** 0x00000000 specifies the logical y-coordinate of the upper-left corner of the source rectangle.

**xformSrc:** 0x3F800000, 0x00000000, 0x00000000, 0x3F800000, 0x00000000, 0x00000000 specifies the world-space to page-space transformation. For more information on coordinate spaces, see [\[MSDN-WRLDPGSPC\]](#).

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
BkColorSrc (0x00FFFFFF)																																	
UsageSrc (0x00000000)																																	
offBmiSrc (0x00000064)																																	
cbBmiSrc (0x00000000)																																	
offBitsSrc (0x0000008C)																																	
cbBitsSrc (0x00002F1C)																																	

**Figure 15: EMR\_BITBLT Record Example, Part 3**

**BkColorSrc:** 0x00FFFFFF specifies the background color RGB value of the source bitmap.

**UsageSrc:** 0x00000000 specifies the value of the **Colors** field of the **WMFDeviceIndependentBitmap** object, specified in [\[MS-WMF\]](#) section 2.2.2.3. The value **MUST** be in the **DIBColors (section 2.1.9)** enumeration.

**offBmiSrc:** 0x00000064 specifies the offset to the source **DeviceIndependentBitmap** object.

**cbBmiSrc:** 0x00000000 specifies the size of the source **DeviceIndependentBitmap** object.

**offBitsSrc:** 0x0000008C specifies the offset to the source bitmap bits.

**cbBitsSrc:** 0x00002F1C specifies the size of the source bitmap bits.

### 3.2.7 EMR\_SETBKMODE Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_SETBKMODE](#) record, specified in section [2.3.11.11](#).

```
00003110:12 00 00 00 0C 00 00 00 01 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000012)																															
Size (0x0000000C)																															
Mode (0x00000001)																															

**Figure 16: Record type EMR\_SETBKMODE example**

**Type:** 0x00000012 identifies this record type as EMR\_SETBKMODE.

**Size:** 0x0000000C specifies the size of this record in bytes.

**Mode:** 0x00000001 specifies the background color value.

### 3.2.8 EMR\_EXTCREATEFONTINDIRECTW Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_EXTCREATEFONTINDIRECTW](#) record, specified in section [2.3.7.8](#).

```
00003110: 52 00 00 00
00003120:70 01 00 00 02 00 00 00 F3 FF FF FF 00 00 00 00
00003130:4E 0C 00 00 4E 0C 00 00 C8 00 00 00 00 00 00 01
00003140:04 00 00 02 41 00 72 00 69 00 61 00 6C 00 00 00
00003150:00 00 00 00 00 00 00 00 00 00 00 00 0C 45 00 00
00003160:BC 16 E8 FE FE 07 00 00 20 00 CC 00 00 00 00 00
00003170:4C 00 00 00 FE 07 00 00 DA 16 01 B8 FF FF FF FF
00003180:00 00 00 00 00 00 00 00 F0 F6 13 00 00 00 00 00
00003190:0E 20 05 27 00 00 00 00 28 00 00 00 00 00 00 00
000031A0:1C 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000031B0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000031C0:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000031D0:00 00 00 00 00 00 00 00 28 00 00 00 FF FF FF 00
000031E0:1C 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000031F0:58 00 00 00 2C 00 00 00 00 00 00 00 00 00 00 00
00003200:00 00 80 3F 00 00 00 00 00 00 00 00 00 00 80 3F
00003210:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003220:D0 C5 35 00 00 00 00 00 28 00 00 00 59 00 00 00
00003230:2D 00 00 00 01 00 18 00 00 00 00 00 1C 2F 00 00
00003240:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003250:10 00 90 01 00 00 00 00 25 00 00 00 00 00 00 00
```

```

00003260:D3 3F EC FE FE 07 00 00 79 0D 21 11 00 00 00 00
00003270:10 00 90 01 00 00 00 00 00 00 00 59 00 00 00
00003280:2D 00 00 00 64 76 00 08 00 00 00 00 00

```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000052)																															
Size (0x00000170)																															
lhFonts (0x00000002)																															
elw (variable)																															
...																															
(elw cont'd for 87 rows)																															

**Figure 17: EMR\_EXTCREATEFONTINDIRECTW Record Example**

**Type:** 0x00000052 identifies this record type as EMR\_EXTCREATEFONTINDIRECTW.

**Size:** 0x00000170 specifies the size of this record in bytes.

**lhFonts:** 0x00000002 specifies the object index in the [EMF Object Table](#) to assign to the font. For more information, see section [3.1.1.1](#).

**elw:** To determine the type of logical font object in this field, the algorithm presented in section [2.3.7.8](#) is applied, which indicates this is a variable-length [LogFontExDv](#) object, specified in section [2.2.13](#).

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Height (0xFFFFFFFF3)																																									
Width (0x00000000)																																									
Escapement (0x00000C4E)																																									
Orientation (0x00000C4E)																																									
Weight (0x000000C8)																																									
Italic (0x00)								Underline (0x00)								StrikeOut (0x00)								CharSet (0x01)																	
OutPrecision (0x04)								ClipPrecision (0x00)								Quality (0x00)								PitchAndFamily (0x02)																	
Facename ("Arial")																																									
...																																									
(Facename cont'd for 14 rows)																																									

**Figure 18: LogFontExDv Object, Part 1**

**Height:** 0xFFFFFFFF3 has an absolute value of 13, which specifies the character height for this font in logical units.

**Width:** 0x00000000 specifies a computed font width. The aspect ratio of the device is matched against the digitization aspect ratio of the font to find the closest match, determined by the absolute value of the difference.

**Escapement:** 0x00000C4E specifies an angle of 315 degrees between the baseline of a row of text and the x-axis of the device.

**Orientation:** 0x00000C4E specifies an angle of 315 degrees between each character's baseline and the x-axis of the device.

**Weight:** 0x000000C8 specifies that the weight of the font is 200, in the range 0 through 1000, from lightest to darkest, with 400 (0x00000190) considered normal.

**Italic:** 0x00 specifies that the font is not italic.

**Underline:** 0x00 specifies that the font is not underlined.

**Strikeout:** 0x00 specifies that the font characters do not have a strike-out graphic.

**CharSet:** 0x01 specifies the default character set, from the [Windows Metafile Format \(WMF\) CharacterSet](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.5**.

**OutPrecision:** 0x04 specifies the output precision, which is how closely the output must match the requested font properties, from the WMF [OutPrecision](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.23**. The value 0x04 specifies that the font mapper should choose a TrueType font, if there is a choice between multiple fonts with the same name.

**ClipPrecision:** 0x00 specifies the clipping precision, which is how to clip characters that are partially outside the clipping region, from the WMF [ClipPrecision](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.6**. The value 0x00 specifies default clipping behavior.

**Quality:** 0x00 specifies default output quality, from the WMF [FontQuality](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.12**.

**PitchAndFamily:** 0x02 specifies a variable-pitch font, and no preference for font family, from the WMF [FamilyFont](#) and [PitchFont](#) enumerations, specified in [\[MS-WMF\]](#) sections **2.1.10** and **2.1.26**, respectively.

**Facename:** "Arial" specifies the typeface name of the font in Unicode characters.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FullName ("")																															
...																															
(FullName cont'd for 30 rows)																															
Style ("")																															
...																															
(Style cont'd for 14 rows)																															
Script ("")																															
...																															
(Script cont'd for 14 rows)																															
Signature (0x80007664)																															
NumAxes (0x00000000)																															

**Figure 19: LogFontExDv Object, Part 2**

**FullName:** An empty string specifies the font's full name.

**Style:** An empty string describes the font's style.

**Script:** An empty string describes the font's character set.

**Signature:** 0x80007664 specifies the signature of an EMF [DesignVector](#) object, specified in section [2.2.2](#).

**NumAxes:** 0x00000000 specifies the number of font axes described in the DesignVector object.

### 3.2.9 EMR\_SELECTOBJECT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_SELECTOBJECT](#) record, specified in section [2.3.8.5](#).

```
00003280:                                     25 00 00 00
00003290:0C 00 00 00 02 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000002)																															

**Figure 20: EMR\_SELECTOBJECT Record Example**

**Type:** 0x00000025 identifies this record type as EMR\_SELECTOBJECT. This value MUST be 0x00000025.

**Size:** 0x0000000C specifies the size of this record in bytes.

**ihObject:** 0x00000002 specifies the index of an object in the [EMF Object Table](#).

**3.2.10 EMR\_EXTTEXTOUTW Example**

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_EXTTEXTOUTW](#) record, specified in section [2.3.5.8](#).

```
00003290:          54 00 00 00 A0 00 00 00
000032A0:00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF
000032B0:01 00 00 00 AB 0A 0D 42 00 00 0D 42 12 00 00 00
000032C0:05 00 00 00 0E 00 00 00 4C 00 00 00 00 00 00 00
000032D0:00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF
000032E0:68 00 00 00 53 00 69 00 6D 00 70 00 6C 00 65 00
000032F0:20 00 53 00 61 00 6D 00 70 00 6C 00 65 00 00 00
00003300:09 00 00 00 03 00 00 00 0B 00 00 00 07 00 00 00
00003310:03 00 00 00 07 00 00 00 04 00 00 00 09 00 00 00
00003320:07 00 00 00 0B 00 00 00 07 00 00 00 03 00 00 00
00003330:07 00 00 00 09 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000054)																															
Size (0x000000A0)																															
Bounds (0x00000000)																															
... (0x00000000)																															
... (0xFFFFFFFF)																															
... (0xFFFFFFFF)																															
iGraphicsMode (0x00000001)																															
exScale (35.260418)																															
eyScale (35.250000)																															
aemrtext (variable)																															

**Figure 21: EMR\_EXTTEXTOUTW Record Example**

**Type:** 0x00000054 identifies this record type as EMR\_EXTTEXTOUTW. This value MUST be 0x00000054.

**Size:** 0x000000A0 specifies the size of this record in bytes.

**Bounds:** 0x00000000, 0x00000000, 0xFFFFFFFF, 0xFFFFFFFF specifies the bounding rectangle in device units.

**iGraphicsMode:** 0x00000001 specifies the current graphics mode.

**exScale:** 35.260418 specifies the X scale from page units to .01mm units if graphics mode is **GM\_COMPATIBLE**.

**eyScale:** 35.250000 specifies the Y scale from page units to .01mm units if graphics mode is **GM\_COMPATIBLE**.

**aemrtext:** An array of [EmrText \(section 2.2.4\)](#) objects that specifies the properties of the strings to be output, and where to find the output strings and spacing values.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reference (0x00000012)																															
... (0x00000005)																															
Chars (0x0000000E)																															
offString (0x0000004C)																															
Options (0x00000000)																															
Rectangle (0x00000000)																															
... (0x00000000)																															
... (0xFFFFFFFF)																															
... (0xFFFFFFFF)																															
offDx (0x00000068)																															
text ("Simple Sample")																															

**Figure 22: EmrText Object Example**

**Reference:** 0x00000012, 0x00000005 specifies the coordinates of the reference point used to position the string.

**Chars:** 0x0000000E specifies the number of characters in the string.

**offString:** 0x0000004C specifies the offset to the string, in bytes from the start of the EMR\_EXTTEXTOUTW record.

**Options:** 0x00000000 specifies that the **Rectangle** field is not used.

**Rectangle:** 0x00000000, 0x00000000, 0xFFFFFFFF, 0xFFFFFFFF values are not used.

**offDx:** 0x00000068 specifies the offset to an intercharacter spacing array, in bytes from the start of the EMR\_EXTTEXTOUTW record.

**text:** "Simple Sample".

### 3.2.11 EMR\_EXTCREATEFONTINDIRECTW Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_EXTCREATEFONTINDIRECTW](#) record, specified in section [2.3.7.8](#).

```
00003330:                52 00 00 00 70 01 00 00
00003340:03 00 00 00 F3 FF FF FF 00 00 00 00 4E 0C 00 00
00003350:4E 0C 00 00 C8 00 00 00 00 00 00 00 04 00 00 02
00003360:4D 00 69 00 63 00 72 00 6F 00 73 00 6F 00 66 00
00003370:74 00 20 00 53 00 61 00 6E 00 73 00 20 00 53 00
```



```

00003380:65 00 72 00 69 00 66 00 00 00 00 00 00 00 00
00003390:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000033A0:00 00 00 00 28 00 00 00 59 00 00 00 2D 00 00 00
000033B0:01 00 18 00 00 00 00 00 1C 2F 00 00 00 00 00 00
000033C0:00 00 00 00 00 00 00 00 00 00 00 00 10 00 90 01
000033D0:00 00 00 00 25 00 00 00 00 00 00 00 D3 3F EC FE
000033E0:FE 07 00 00 79 0D 21 11 00 00 00 00 10 00 90 01
000033F0:00 00 00 00 00 00 00 00 59 00 00 00 2D 00 00 00
00003400:64 76 00 08 00 00 00 00 00 00 00 00 00 08 0D 00
00003410:0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003420:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003430:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003440:00 00 00 00 92 A0 CD 02 00 00 00 00 CA BE CD 02
00003450:00 00 00 00 00 00 00 00 00 00 00 00 FF FF 5A FE
00003460:00 00 00 00 00 00 00 00 00 00 00 00 95 F1 53 FE
00003470:FE 07 00 00 BE 06 5A FE FE 07 00 00 87 F2 53 FE
00003480:FE 07 00 00 C4 04 5A FE FE 07 00 00 79 0D 21 11
00003490:00 00 00 00 01 00 00 00 00 00 00 00 F0 02 5A FE
000034A0:64 76 00 08 00 00 00 00

```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000052)																															
Size (0x00000170)																															
ihFonts (0x00000003)																															
elw (variable)																															
...																															
(elw cont'd for 87 rows)																															

**Figure 23: EMR\_EXTCREATEFONTINDIRECTW Record Example**

**Type:** 0x00000052 identifies this record type as EMR\_EXTCREATEFONTINDIRECTW.

**Size:** 0x00000170 specifies the size of this record in bytes.

**ihFonts:** 0x00000003 specifies the object index in the [EMF Object Table](#) to assign to the font. For more information, see section [3.1.1.1](#).

**elw:** To determine the type of logical font object in this field, the algorithm presented in section [2.3.7.8](#) is applied, which indicates this is a variable-length [LogFontExDv](#) object, specified in section [2.2.13](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Height (0xFFFFFFFF3)																															
Width (0x00000000)																															
Escapement (0x00000C4E)																															
Orientation (0x00000C4E)																															
Weight (0x000000C8)																															
Italic (0x00)								Underline (0x00)								StrikeOut (0x00)								CharSet (0x00)							
OutPrecision (0x04)								ClipPrecision (0x00)								Quality (0x00)								PitchAndFamily (0x02)							
Facename ("Mircosoft Sans Serif")																															
...																															
(Facename cont'd for 14 rows)																															

**Figure 24: LogFontExDv Object, Part 1**

**Height:** 0xFFFFFFFF3 has an absolute value of 13, which specifies the character height for this font in logical units.

**Width:** 0x00000000 specifies a computed font width. The aspect ratio of the device is matched against the digitization aspect ratio of the font to find the closest match, determined by the absolute value of the difference.

**Escapement:** 0x00000C4E specifies an angle of 315 degrees between the baseline of a row of text and the x-axis of the device.

**Orientation:** 0x00000C4E specifies an angle of 315 degrees between each character's baseline and the x-axis of the device.

**Weight:** 0x000000C8 specifies that the weight of the font is 200, in the range 0 through 1000, from lightest to darkest, with 400 (0x00000190) considered normal.

**Italic:** 0x00 specifies that the font is not italic.

**Underline:** 0x00 specifies that the font is not underlined.

**Strikeout:** 0x00 specifies that the font characters do not have a strike-out graphic.

**CharSet:** 0x00 specifies the ANSI character set, from the [Windows Metafile Format \(WMF\) CharacterSet](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.5**.

**OutPrecision:** 0x04 specifies the output precision, which is how closely the output must match the requested font properties, from the WMF [OutPrecision](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.23**. The value 0x04 specifies that the font mapper should choose a TrueType font, if there is a choice between multiple fonts with the same name..

**ClipPrecision:** 0x00 specifies the clipping precision, which is how to clip characters that are partially outside the clipping region, from the WMF [ClipPrecision](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.6**. The value 0x00 specifies default clipping behavior.

**Quality:** 0x00 specifies default output quality, from the WMF [FontQuality](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.12**.

**PitchAndFamily:** 0x02 specifies a variable-pitch font, and no preference for font family, from the WMF [FamilyFont](#) and [PitchFont](#) enumerations, specified in [\[MS-WMF\]](#) sections **2.1.10** and **2.1.26**, respectively.

**Facename:** "Microsoft Sans Serif" specifies the typeface name of the font in Unicode characters.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
FullName ("")																															
...																															
(FullName cont'd for 30 rows)																															
Style ("")																															
...																															
(Style cont'd for 14 rows)																															
Script ("")																															
...																															
(Script cont'd for 14 rows)																															
Signature (0x80007664)																															
NumAxes (0x00000000)																															

**Figure 25: LogFontExDv Object, Part 2**

**FullName:** An empty string specifies the font's full name.

**Style:** An empty string describes the font's style.

**Script:** An empty string describes the font's character set.

**Signature:** 0x80007664 specifies the signature of an EMF [DesignVector](#) object, specified in section [2.2.2](#).

**NumAxes:** 0x00000000 specifies the number of font axes described in the DesignVector object.

**3.2.12 EMR\_SELECTOBJECT Example**

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_SELECTOBJECT](#) record, specified in section [2.3.8.5](#).

```
000034A0:                25 00 00 00 0C 00 00 00
000034B0:03 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000003)																															

**Figure 26: EMR\_SELECTOBJECT Record Example**

**Type:** Identifies this record type as EMR\_SELECTOBJECT. This value MUST be 0x00000025.

**Size:** Specifies the size of this record in bytes. This value MUST be 0x0000000C.

**ihObject:** Specifies the index of an object in the [EMF Object Table](#). This value MUST be 0x00000003.

### 3.2.13 EMR\_EXTCREATEFONTINDIRECTW Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_EXTCREATEFONTINDIRECTW](#) record, specified in section [2.3.7.8](#).

```

000034B0:          52 00 00 00 70 01 00 00 04 00 00 00
000034C0:F2 FF FF FF 00 00 00 00 4E 0C 00 00 4E 0C 00 00
000034D0:C8 00 00 00 00 00 00 00 04 00 00 02 4D 00 69 00
000034E0:63 00 72 00 6F 00 73 00 6F 00 66 00 74 00 20 00
000034F0:53 00 61 00 6E 00 73 00 20 00 53 00 65 00 72 00
00003500:69 00 66 00 00 00 00 00 00 00 00 00 00 00 00 00
00003510:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00003520:C8 F1 13 00 00 00 00 00 10 00 90 01 00 00 00 00
00003530:20 AC CF 02 00 00 00 00 10 00 00 00 06 00 00 00
00003540:06 00 00 00 04 00 00 00 01 00 00 00 01 00 00 00
00003550:01 00 00 00 01 00 00 00 0D 00 00 00 00 00 00 00
00003560:03 00 00 00 00 08 00 00 3B 09 00 00 00 00 00 00
00003570:A0 E8 07 02 00 00 00 00 03 00 00 00 FE 07 00 00
00003580:90 01 00 00 4D 00 69 00 63 00 72 00 6F 00 73 00
00003590:6F 00 66 00 74 00 20 00 53 00 61 00 00 00 73 00
000035A0:20 00 53 00 65 00 72 00 69 00 66 00 00 00 00 00
000035B0:00 00 00 00 00 00 00 00 FF FF 5A FE 00 00 00 00
000035C0:40 02 5A FE FE 07 00 00 9D 04 00 00 00 00 00 00
000035D0:FF FF FF FF FF FF FF 01 00 00 00 00 00 00 00
000035E0:20 AC CF 02 00 00 00 00 00 00 07 02 00 00 00 00
000035F0:10 AC CF 02 00 00 00 00 26 36 E3 76 00 00 00 00
00003600:00 00 07 02 00 00 00 00 01 00 00 00 00 27 00 00
00003610:00 00 00 00 00 00 00 00 20 AC CF 02 64 76 00 08
00003620:00 00 00 00

```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000052)																															
Size (0x00000170)																															
ihFonts (0x00000004)																															
elw (variable)																															
...																															
(elw cont'd for 87 rows)																															

**Figure 27: EMR\_EXTCREATEFONTINDIRECTW Record Example**

**Type:** 0x00000052 identifies this record type as EMR\_EXTCREATEFONTINDIRECTW.

**Size:** 0x00000170 specifies the size of this record in bytes.

**ihFonts:** 0x00000004 specifies the object index in the [EMF Object Table](#) to assign to the font. For more information, see section [3.1.1.1](#).

**elw:** To determine the type of logical font object in this field, the algorithm presented in section [2.3.7.8](#) is applied, which indicates this is a variable-length [LogFontExDv](#) object, specified in section [2.2.13](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Height (0xFFFFFFFF2)																															
Width (0x00000000)																															
Escapement (0x00000C4E)																															
Orientation (0x00000C4E)																															
Weight (0x000000C8)																															
Italic (0x00)								Underline (0x00)								StrikeOut (0x00)								CharSet (0x00)							
OutPrecision (0x04)								ClipPrecision (0x00)								Quality (0x00)								PitchAndFamily (0x02)							
Facename ("Microsoft Sans Serif")																															
...																															
(Facename cont'd for 14 rows)																															

**Figure 28: LogFontExDv Object, Part 1**

**Height:** 0xFFFFFFFF2 has an absolute value of 14, which specifies the character height for this font in logical units.

**Width:** 0x00000000 specifies a computed font width. The aspect ratio of the device is matched against the digitization aspect ratio of the font to find the closest match, determined by the absolute value of the difference.

**Escapement:** 0x00000C4E specifies an angle of 315 degrees between the baseline of a row of text and the x-axis of the device.

**Orientation:** 0x00000C4E specifies an angle of 315 degrees between each character's baseline and the x-axis of the device.

**Weight:** 0x000000C8 specifies that the weight of the font is 200, in the range 0 through 1000, from lightest to darkest, with 400 (0x00000190) considered normal.

**Italic:** 0x00 specifies that the font is not italic.

**Underline:** 0x00 specifies that the font is not underlined.

**Strikeout:** 0x00 specifies that the font characters do not have a strike-out graphic.

**CharSet:** 0x00 specifies the ANSI character set, from the [Windows Metafile Format \(WMF\) CharacterSet](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.5**.

**OutPrecision:** 0x04 specifies the output precision, which is how closely the output must match the requested font properties, from the WMF [OutPrecision](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.23**. The value 0x04 specifies that the font mapper should choose a TrueType font, if there is a choice between multiple fonts with the same name.

**ClipPrecision:** 0x00 specifies the clipping precision, which is how to clip characters that are partially outside the clipping region, from the WMF [ClipPrecision](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.6**. The value 0x00 specifies default clipping behavior.

**Quality:** 0x00 specifies default output quality, from the WMF [FontQuality](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.12**.

**PitchAndFamily:** 0x02 specifies a variable-pitch font, and no preference for font family, from the WMF [FamilyFont](#) and [PitchFont](#) enumerations, specified in [\[MS-WMF\]](#) sections **2.1.10** and **2.1.26**, respectively.

**Facename:** "Microsoft Sans Serif" specifies the typeface name of the font in Unicode characters.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
FullName ("")																																		
...																																		
(FullName cont'd for 30 rows)																																		
Style ("")																																		
...																																		
(Style cont'd for 14 rows)																																		
Script ("")																																		
...																																		
(Script cont'd for 14 rows)																																		
Signature (0x80007664)																																		
NumAxes (0x00000000)																																		

## Figure 29: LogFontExDv Object, Part 2

**FullName:** An empty string specifies the font's full name.

**Style:** An empty string describes the font's style.

**Script:** An empty string describes the font's character set.

**Signature:** 0x80007664 specifies the signature of an EMF [DesignVector](#) object, specified in section [2.2.2](#).

**NumAxes:** 0x00000000 specifies the number of font axes described in the DesignVector object.

### 3.2.14 EMR\_SELECTOBJECT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_SELECTOBJECT](#) record, specified in section [2.3.8.5](#).

00003620: 25 00 00 00 0C 00 00 00 04 00 00 00

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000004)																															

Figure 30: Record type EMR\_SELECTOBJECT example

**Type:** Identifies this record type as EMR\_SELECTOBJECT. This value MUST be 0x00000025.

**Size:** Specifies the size of this record in bytes. This value MUST be 0x0000000C.

**ihObject:** Specifies the index of an object in the [EMF Object Table](#). This value MUST be 0x00000004.

### 3.2.15 EMR\_DELETEOBJECT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_DELETEOBJECT](#) record, specified in section [2.3.8.3](#).

00003630: 28 00 00 00 0C 00 00 00 03 00 00 00

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000028)																															
Size (0x0000000C)																															
ihObject (0x00000003)																															

Figure 31: EMR\_DELETEOBJECT Record Example

**Type:** Identifies this record type as EMR\_DELETEOBJECT. This value MUST be 0x00000028.

**Size:** Specifies the size of this record in bytes. This value MUST be 0x0000000C.

**ihObject:** Specifies the index of the [EMF Object Table](#) object to be deleted. This value MUST be 0x00000003.

### 3.2.16 EMR\_EXTCREATEFONTINDIRECTW Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_EXTCREATEFONTINDIRECTW](#) record, specified in section [2.3.7.8](#).

```
00003630:                                     52 00 00 00
00003640:70 01 00 00 03 00 00 00 13 00 00 00 00 00 00 00
00003650:4E 0C 00 00 4E 0C 00 00 C8 00 00 00 00 00 00 00
00003660:04 00 00 02 4D 00 69 00 63 00 72 00 6F 00 73 00
00003670:6F 00 66 00 74 00 20 00 53 00 61 00 6E 00 73 00
00003680:20 00 53 00 65 00 72 00 69 00 66 00 00 00 00 00
00003690:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000036A0:00 00 00 00 00 00 73 00 20 00 53 00 65 00 72 00
000036B0:69 00 66 00 00 00 00 00 00 00 00 00 00 00 00 00
000036C0:FF FF 5A FE 00 00 00 00 40 02 5A FE FE 07 00 00
000036D0:9D 04 00 00 00 00 00 00 FF FF FF FF FF FF FF FF
000036E0:01 00 00 00 00 00 00 00 20 AC CF 02 00 00 00 00
000036F0:00 00 07 02 00 00 00 00 40 02 5A FE FE 07 00 00
00003700:F3 14 00 00 00 00 00 00 F3 14 0A 1E 00 00 00 00
00003710:94 8A E8 FE FE 07 00 00 04 00 00 00 00 00 00 00
00003720:65 58 53 FE 00 00 00 00 00 00 00 00 00 00 00 00
00003730:00 F5 13 00 00 00 00 00 03 01 56 E5 89 1A 00 00
00003740:55 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00
00003750:00 00 00 00 FE 07 00 00 79 0D 21 11 00 00 00 00
00003760:40 02 5A FE 00 00 00 00 26 06 5A FE FE 07 00 00
00003770:08 F5 13 00 00 00 00 00 00 F5 13 00 00 00 00 00
00003780:07 CB 54 FE FE 07 00 00 79 0D 21 11 00 00 00 00
00003790:04 00 00 00 00 00 00 00 24 07 5A FE FE 07 00 00
000037A0:01 00 00 00 64 76 00 08 00 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000052)																															
Size (0x00000170)																															
lhFonts (0x00000003)																															
elw (variable)																															
...																															
(elw cont'd for 87 rows)																															

**Figure 32: EMR\_EXTCREATEFONTINDIRECTW Record Example**

**Type:** 0x00000052 identifies this record type as EMR\_EXTCREATEFONTINDIRECTW.

**Size:** 0x00000170 specifies the size of this record in bytes.



**ihFonts:** 0x00000003 specifies the object index in the [EMF Object Table](#) to assign to the font. For more information, see section [3.1.1.1](#).

**elw:** To determine the type of logical font object in this field, the algorithm presented in section [2.3.7.8](#) is applied, which indicates this is a variable-length [LogFontExDv](#) object, specified in section [2.2.13](#).

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Height (0x00000013)																															
Width (0x00000000)																															
Escapement (0x00000C4E)																															
Orientation (0x00000C4E)																															
Weight (0x000000C8)																															
Italic (0x00)								Underline (0x00)								StrikeOut (0x00)								CharSet (0x01)							
OutPrecision (0x04)								ClipPrecision (0x00)								Quality (0x00)								PitchAndFamily (0x02)							
Facename ("Mircosoft Sans Serif")																															
...																															
(Facename cont'd for 14 rows)																															

**Figure 33: LogFontExDv Object, Part 1**

**Height:** 0x00000013 specifies the cell height for this font in logical units.

**Width:** 0x00000000 specifies a computed font width. The aspect ratio of the device is matched against the digitization aspect ratio of the font to find the closest match, determined by the absolute value of the difference.

**Escapement:** 0x00000C4E specifies an angle of 315 degrees between the baseline of a row of text and the x-axis of the device.

**Orientation:** 0x00000C4E specifies an angle of 315 degrees between each character's baseline and the x-axis of the device.

**Weight:** 0x000000C8 specifies that the weight of the font is 200, in the range 0 through 1000, from lightest to darkest, with 400 (0x00000190) considered normal.

**Italic:** 0x00 specifies that the font is not italic.

**Underline:** 0x00 specifies that the font is not underlined.

**Strikeout:** 0x00 specifies that the font characters do not have a strike-out graphic.

**CharSet:** 0x00 specifies the ANSI character set, from the [Windows Metafile Format \(WMF\) CharacterSet](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.5**.

**OutPrecision:** 0x04 specifies the output precision, which is how closely the output must match the requested font properties, from the WMF [OutPrecision](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.23**. The value 0x04 specifies that the font mapper should choose a TrueType font, if there is a choice between multiple fonts with the same name.

**ClipPrecision:** 0x00 specifies the clipping precision, which is how to clip characters that are partially outside the clipping region, from the WMF [ClipPrecision](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.6**. The value 0x00 specifies default clipping behavior.

**Quality:** 0x00 specifies default output quality, from the WMF [FontQuality](#) enumeration, specified in [\[MS-WMF\]](#) section **2.1.12**.

**PitchAndFamily:** 0x02 specifies a variable-pitch font, and no preference for font family, from the WMF [FamilyFont](#) and [PitchFont](#) enumerations, specified in [\[MS-WMF\]](#) sections **2.1.10** and **2.1.26**, respectively.

**Facename:** "Microsoft Sans Serif" specifies the typeface name of the font in Unicode characters.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FullName ("")																															
...																															
(FullName cont'd for 30 rows)																															
Style ("")																															
...																															
(Style cont'd for 14 rows)																															
Script ("")																															
...																															
(Script cont'd for 14 rows)																															
Signature (0x80007664)																															
NumAxes (0x00000000)																															

**Figure 34: LogFontExDv Object, Part 2**

**FullName:** An empty string specifies the font's full name.

**Style:** An empty string describes the font's style.

**Script:** An empty string describes the font's character set.

**Signature:** 0x80007664 specifies the signature of an EMF [DesignVector](#) object, specified in section [2.2.2](#).

**NumAxes:** 0x00000000 specifies the number of font axes described in the DesignVector object.

**3.2.17 EMR\_SELECTOBJECT Example**

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_SELECTOBJECT](#) record, specified in section [2.3.8.5](#).

```
000037A0:                                     25 00 00 00
000037B0:0c 00 00 00 03 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000003)																															

**Figure 35: EMR\_SELECTOBJECT Record Example**

**Type:** 0x00000025 identifies this record type as EMR\_SELECTOBJECT. This value MUST be 0x00000025.

**Size:** 0x0000000C specifies the size of this record in bytes.

**ihObject:** 0x00000003 specifies the index of an object in the [EMF Object Table](#).

### 3.2.18 EMR\_SELECTOBJECT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_SELECTOBJECT](#) record, specified in section [2.3.8.5](#).

```
000037B0:                25 00 00 00 0C 00 00 00
000037C0:02 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x00000002)																															

**Figure 36: EMR\_SELECTOBJECT Record Example**

**Type:** 0x00000025 identifies this record type as EMR\_SELECTOBJECT. This value MUST be 0x00000025.

**Size:** 0x0000000C specifies the size of this record in bytes.

**ihObject:** 0x00000002 specifies the index of an object in the [EMF Object Table](#).

### 3.2.19 EMR\_DELETEOBJECT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_DELETEOBJECT](#) record, specified in section [2.3.8.3](#).

```
000037C0:                28 00 00 00 0C 00 00 00 04 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000028)																															
Size (0x0000000C)																															
ihObject (0x00000004)																															

**Figure 37: EMR\_DELETEOBJECT Record Example**

**Type:** 0x00000028 identifies this record type as EMR\_DELETEOBJECT. This value MUST be 0x00000028.

**Size:** 0x0000000C specifies the size of this record in bytes.

**ihObject:** 0x00000004 specifies the index of the [EMF Object Table](#) object to be deleted.

### 3.2.20 EMR\_DELETEOBJECT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_DELETEOBJECT](#) record, specified in section [2.3.8.3](#).

```
000037D0:28 00 00 00 0C 00 00 00 03 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000028)																															
Size (0x0000000C)																															
ihObject (0x00000003)																															

**Figure 38: EMR\_DELETEOBJECT Record Example**

**Type:** 0x00000028 identifies this record type as EMR\_DELETEOBJECT. This value MUST be 0x00000028.

**Size:** 0x0000000C specifies the size of this record in bytes.

**ihObject:** 0x00000003 specifies the index of the [EMF Object Table](#) object to be deleted.

### 3.2.21 EMR\_SELECTOBJECT Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_SELECTOBJECT](#) record, specified in section [2.3.8.5](#).

```
000037D0:                25 00 00 00
000037E0:0C 00 00 00 0D 00 00 80
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x00000025)																															
Size (0x0000000C)																															
ihObject (0x8000000D = SYSTEM_FONT)																															

**Figure 39: EMR\_SELECTOBJECT Record Example**

**Type:** 0x00000025 identifies this record type as EMR\_SELECTOBJECT. This value MUST be 0x00000025.

**Size:** 0x0000000C specifies the size of this record in bytes.

**ihObject:** 0x8000000D specifies the index of an object in the [EMF Object Table](#).

### 3.2.22 EMR\_EOF Example

This section provides an example of the Enhanced Metafile Format (EMF) [EMR\\_EOF](#) record, specified in section [2.3.4.1](#).

```
000037E0:                                0E 00 00 00 14 00 00 00
000037F0:00 00 00 00 10 00 00 00 14 00 00 00
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Type (0x0000000E)																															
Size (0x00000014)																															
nPalEntries (0x00000000)																															
offPalEntries (0x00000010)																															
SizeLast (0x00000014)																															

**Figure 40: EMR\_EOF Record Example**

**Type:** 0x0000000E identifies this record type as EMR\_EOF. This value MUST be 0x0000000E.

**Size:** 0x00000014 specifies the size of this record in bytes.

**nPalEntries:** 0x00000000 specifies the number of palette entries.

**offPalEntries:** 0x00000010 specifies the offset to the palette entries.

**SizeLast:** 0x00000014 is the same as size and MUST be the last unsigned 32-bit integer of the record. The palette entries, if they exist, MUST precede this field.

## 4 Security Considerations

This file format enables third parties to send payloads (such as PostScript) to pass through as executable code.

## 5 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows Server 2008
- Windows Vista
- Windows Server 2003
- Windows XP
- Windows 2000
- Windows Me
- Windows NT Workstation
- Windows 98
- Windows NT 4.0
- Windows 95

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

[<1> Section 1.3.1:](#) Enhanced Metafile Format metafiles have evolved over the years, with the evolution of Windows operating systems. They were introduced with 32-bit versions, and replaced the older, 16-bit Windows Metafile Format as the standard, as specified in [\[MS-WMF\]](#).

[<2> Section 1.3.1:](#) This form of Enhanced Metafile Format metafile has been supported starting with Windows 95 and Windows NT 4.0. It was the first form of Windows metafile that supported device-independence.

[<3> Section 1.3.1:](#) This form of Enhanced Metafile Format metafile has been supported starting with Windows 98.

[<4> Section 1.3.1:](#) This form of Enhanced Metafile Format metafile has been supported starting with Windows 2000.

[<5> Section 1.4:](#) Windows applications that use the **Windows graphics device interface (GDI)** API create metafiles in this format. Windows applications that use the **Windows graphics device interface Plus Extensions (GDI+)** API can create metafiles in either the EMF or [Enhanced Metafile Format Plus Extensions](#) format, or both. For more information, see [\[MS-EMFPLUS\]](#).

The EMF format supersedes WMF, as specified in [\[MS-WMF\]](#), which was used in 16-bit Windows operating systems.

A variant of the EMF format is used as the printer **spool file** format in some Windows operating systems, as specified in [\[MS-EMFSPOOL\]](#).

[<6> Section 1.6:](#) Enhanced Metafile Format support started with Windows 95 and Windows NT 4.0.

[<7> Section 1.6:](#) Enhanced Metafile Format extension 1 support started with Windows 98.

[<8> Section 1.6:](#) Enhanced Metafile Format extension 2 support started with Windows 2000.

[<9> Section 2.1.1:](#) In Windows implementations, **BS\_HOLLOW** was added as a duplicate symbolic name for **BS\_NULL**, in order for implementers not to confuse it with a null pointer to a brush.

Historically, it was used to satisfy the Windows Graphics Device Interface (GDI) requirement for a brush, in situations when the GDI application did not require any kind of brush to be used.

[<10> Section 2.1.11:](#) In Windows, the **ETO\_GLYPH\_INDEX** flag can be used for bitmap and vector fonts—in addition to TrueType fonts—to indicate that no further language processing is necessary, and that the **Windows Graphics Device Interface (GDI)** should process the string directly.

[<11> Section 2.1.11:](#) In Windows, if this bit is set, international scripting support is not used, which may cause no text to be output.

[<12> Section 2.1.16:](#) **GM\_COMPATIBLE** graphics mode is used for compatibility between both 16-bit and 32-bit systems.

[<13> Section 2.1.16:](#) Windows 95, Windows 98, and Windows Me: **GM\_ADVANCED** is not supported.

[<14> Section 2.1.22:](#) In Windows, this value indicates the version of operating system that was used to create the metafile.

[<15> Section 2.1.22:](#) In Windows, this value indicates the version of operating system that was used to create the metafile.

[<16> Section 2.1.31:](#) In Windows implementations, **HOLLOW\_BRUSH** was added as a duplicate symbolic name for **NULL\_BRUSH**, in order for implementers not to confuse it with a null pointer to a brush.

Historically, it was used to satisfy the Windows Graphics Device Interface (GDI) requirement for a brush, in situations when the GDI application did not require any kind of brush to be used.

A null brush paints nothing.

[<17> Section 2.1.31:](#) This is typically the "Courier" font.

[<18> Section 2.1.31:](#) This is typically the "MS Sans Serif" font.

[<19> Section 2.1.31:](#) This is the font used by Windows to draw menu text and dialog box controls.

Windows 95, Windows 98 and Windows NT: The system font is "MS Sans Serif".

Windows 2000, Windows XP, Windows Vista, and Windows Server 2008: The system font is "Tahoma".

[<20> Section 2.1.31:](#) Windows 2000, Windows NT, Windows XP, Windows Vista, and Windows Server 2008: A font installed on a device.

[<21> Section 2.1.31:](#) Windows 2000, Windows XP, Windows Vista, and Windows Server 2008: The default color is white.

[<22> Section 2.1.31:](#) Windows 2000, Windows XP, Windows Vista, and Windows Server 2008: The default color is white.



<23> [Section 2.1.32:](#) Windows also uses the following symbols for the [StretchMode](#) enumeration; their meanings and operation are exactly the same as specified in section [2.1.32](#):

```
#define BLACKONWHITE          1
#define WHITEONBLACK          2
#define COLORONCOLOR          3
#define HALFTONE               4
```

<24> [Section 2.2.10:](#) In Windows implementations, **BS\_HOLLOW** was added as a duplicate symbolic name for **BS\_NULL**, in order for implementers not to confuse it with a null pointer to a brush.

Historically, it was used to satisfy the **Windows Graphics Device Interface (GDI)** requirement for a brush in situations where the GDI application did not require any kind of brush to be used.

<25> [Section 2.2.11:](#) In Windows implementations, the aspect ratio of the device is matched against the digitization aspect ratios of the available fonts to find the closest match, determined by the absolute value of the difference.

<26> [Section 2.2.11:](#) Windows uses the value 400 by default.

<27> [Section 2.2.18:](#) In Windows implementations, **BS\_HOLLOW** was added as a duplicate symbolic name for **BS\_NULL**, in order for implementers not to confuse it with a null pointer to a brush.

Historically, it was used to satisfy the Windows Graphics Device Interface (GDI) requirement for a brush in situations where the GDI application did not require any kind of brush to be used.

<28> [Section 2.2.20:](#) This flag is not supported.

<29> [Section 2.2.20:](#) If set, the pixel format is supported by GDI.

<30> [Section 2.2.20:](#) Alpha bitplanes are not supported.

<31> [Section 2.2.20:](#) Alpha bitplanes are not supported.

<32> [Section 2.2.20:](#) Alpha bitplanes are not supported.

<33> [Section 2.2.20:](#) This field was used by implementations of OpenGL in Windows 95. See [\[OPENGL\]](#) for more information.

<34> [Section 2.2.20:](#) This field was used by implementations of OpenGL in Windows 95. See [\[OPENGL\]](#) for more information.

<35> [Section 2.2.20:](#) This field was used by implementations of OpenGL in Windows 95. See [\[OPENGL\]](#) for more information.

<36> [Section 2.3.1.8:](#) If the source bitmap color format is 32 bits per pixel, only the alpha transparency data is copied to the destination.

<37> [Section 2.3.3.1:](#) Windows NT 3.1, Windows NT 3.51, Windows 95, and Windows NT 4.0 ignore Enhanced Metafile Format (EMF) [EMR\\_COMMENT](#) records.

<38> [Section 2.3.3.4.3:](#) On playback, the first format recognized by Windows is used to render the image.

[<39> Section 2.3.4.2.2:](#) The Windows implementation determines whether an optional header object is present from the value of the **Size** field in the **EMR\_HEADER** record, specified in section [2.3.4.2](#).

[<40> Section 2.3.4.2.3:](#) The Windows implementation determines whether an optional header object is present from the value of the **Size** field in the **EMR\_HEADER** record, specified in section [2.3.4.2](#).

[<41> Section 2.3.7.2:](#) Windows NT 3.1 and Windows NT 3.51: Not supported.

[<42> Section 2.3.7.3:](#) Windows NT 3.1, Windows NT 3.51, Windows NT 4.0, and Windows 95: Not supported.

[<43> Section 2.3.11.1:](#) Before processing this record, Windows Color System (WCS) MUST be enabled for both the playback device context and target device context. Only Windows Vista and Windows Server 2008 support WCS.

## 6 Index

### A

[Applicability](#)  
[ArcDirection enumeration](#)  
[ArmStyle enumeration](#)

### B

[BackgroundMode enumeration](#)  
[BitmapRecordTypes packet](#)  
[Byte ordering](#)  
[Byte ordering example](#)

### C

[ClippingRecordTypes packet](#)  
[ColorAdjustment enumeration](#)  
[ColorAdjustment packet](#)  
[ColorMatchToTarget enumeration](#)  
[ColorSpace enumeration](#)  
[CommentRecordTypes packet](#)  
[Contrast enumeration](#)  
[ControlRecordTypes packet](#)

### D

[DesignVector packet](#)  
[DIBColors enumeration](#)  
[Drawing Record Types](#)  
[DrawingRecordTypes packet](#)

### E

[EMF metafile example](#)  
[EmfMetafileHeader packet](#)  
[EmfMetafileHeaderExtension1 packet](#)  
[EmfMetafileHeaderExtension2 packet](#)  
[EMR\\_ALPHABLEND packet](#)  
[EMR\\_ANGLEARC packet](#)  
[EMR\\_ARC packet](#)  
[EMR\\_ARCTO packet](#)  
[EMR\\_BITBLT example \(section 3.2.4, section 3.2.6\)](#)  
[EMR\\_BITBLT packet](#)  
[EMR\\_CHORD packet](#)  
[EMR\\_COLORCORRECTPALETTE packet](#)  
[EMR\\_COLORMATCHTOTARGETW packet](#)  
[EMR\\_COMMENT packet](#)  
[EMR\\_COMMENT\\_BEGINGROUP packet](#)  
[EMR\\_COMMENT\\_EMFPLUS packet](#)  
[EMR\\_COMMENT\\_EMFSPOOL packet](#)  
[EMR\\_COMMENT\\_ENDGROUP packet](#)  
[EMR\\_COMMENT\\_MULTIFORMATS packet](#)  
[EMR\\_COMMENT\\_PUBLIC packet](#)  
[EMR\\_COMMENT\\_WINDOWS\\_METAFILE packet](#)  
[EMR\\_CREATEBRUSHINDIRECT example](#)  
[EMR\\_CREATEBRUSHINDIRECT packet](#)  
[EMR\\_CREATECOLORSPACE packet](#)  
[EMR\\_CREATECOLORSPACEW packet](#)  
[EMR\\_CREATEDIBPATTERNBRUSHPT packet](#)

[EMR\\_CREATEMONOBRUSH packet](#)  
[EMR\\_CREATEPALETTE packet](#)  
[EMR\\_CREATEPEN packet](#)  
[EMR\\_DELETECOLORSPACE packet](#)  
[EMR\\_DELETEOBJECT example \(section 3.2.15, section 3.2.19, section 3.2.20\)](#)  
[EMR\\_DELETEOBJECT packet](#)  
[EMR\\_DRAWESCAPE packet](#)  
[EMR\\_ELLIPSE packet](#)  
[EMR\\_EOF example](#)  
[EMR\\_EOF packet](#)  
[EMR\\_EXCLUDECLIPRECT packet](#)  
[EMR\\_EXTCREATEFONTINDIRECTW example \(section 3.2.8, section 3.2.11, section 3.2.13, section 3.2.16\)](#)  
[EMR\\_EXTCREATEFONTINDIRECTW packet](#)  
[EMR\\_EXTCREATEPEN packet](#)  
[EMR\\_EXTESCAPE packet](#)  
[EMR\\_EXTFLOODFILL packet](#)  
[EMR\\_EXTSELECTCLIPRGN packet](#)  
[EMR\\_EXTTEXTOUTA packet](#)  
[EMR\\_EXTTEXTOUTW example](#)  
[EMR\\_EXTTEXTOUTW packet](#)  
[EMR\\_FILLPATH packet](#)  
[EMR\\_FILLRGN packet](#)  
[EMR\\_FORCEUFIMAPPING packet](#)  
[EMR\\_FRAMERGN packet](#)  
[EMR\\_GLSBOUNDEDRECORD packet](#)  
[EMR\\_GLSRECORD packet](#)  
[EMR\\_GRADIENTFILL packet](#)  
[EMR\\_HEADER example](#)  
[EMR\\_HEADER packet](#)  
[EMR\\_INTERSECTCLIPRECT packet](#)  
[EMR\\_INVERTRGN packet](#)  
[EMR\\_LINETO packet](#)  
[EMR\\_MASKBLT packet](#)  
[EMR\\_MODIFYWORLDTRANSFORM packet](#)  
[EMR\\_MOVETOEX packet](#)  
[EMR\\_NAMEDESCAPE packet](#)  
[EMR\\_OFFSETCLIPRGN packet](#)  
[EMR\\_PAINTRGN packet](#)  
[EMR\\_PIE packet](#)  
[EMR\\_PIXELFORMAT packet](#)  
[EMR\\_PLGBLT packet](#)  
[EMR\\_POLYBEZIER packet](#)  
[EMR\\_POLYBEZIER16 packet](#)  
[EMR\\_POLYBEZIERTO packet](#)  
[EMR\\_POLYBEZIERTO16 packet](#)  
[EMR\\_POLYDRAW packet](#)  
[EMR\\_POLYDRAW16 packet](#)  
[EMR\\_POLYGON packet](#)  
[EMR\\_POLYGON16 packet](#)  
[EMR\\_POLYLINE packet](#)  
[EMR\\_POLYLINE16 packet](#)  
[EMR\\_POLYLINETO packet](#)  
[EMR\\_POLYLINETO16 packet](#)  
[EMR\\_POLYPOLYGON packet](#)  
[EMR\\_POLYPOLYGON16 packet](#)  
[EMR\\_POLYPOLYLINE packet](#)  
[EMR\\_POLYPOLYLINE16 packet](#)

[EMR\\_POLYTEXTOUTA packet](#)  
[EMR\\_POLYTEXTOUTW packet](#)  
[EMR\\_RECTANGLE packet](#)  
[EMR\\_RESIZEPALETTE packet](#)  
[EMR\\_RESTOREDC packet](#)  
[EMR\\_ROUNDRECT packet](#)  
[EMR\\_SCALEVIEWPORTEXT packet](#)  
[EMR\\_SCALEWINDOWEXT packet](#)  
[EMR\\_SELECTCLIPPATH packet](#)  
[EMR\\_SELECTOBJECT example \(section 3.2.3, section 3.2.5, section 3.2.9, section 3.2.12, section 3.2.14, section 3.2.17, section 3.2.18, section 3.2.21\)](#)  
[EMR\\_SELECTOBJECT packet](#)  
[EMR\\_SELECTPALETTE packet](#)  
[EMR\\_SETARCDIRECTION packet](#)  
[EMR\\_SETBKCOLOR packet](#)  
[EMR\\_SETBKMODE example](#)  
[EMR\\_SETBKMODE packet](#)  
[EMR\\_SETBRUSHORGEX packet](#)  
[EMR\\_SETCOLORADJUSTMENT packet](#)  
[EMR\\_SETCOLORSPACE packet](#)  
[EMR\\_SETDIBITSTODEVICE packet](#)  
[EMR\\_SETICMMODE packet](#)  
[EMR\\_SETICMPROFILEA packet](#)  
[EMR\\_SETICMPROFILEW packet](#)  
[EMR\\_SETLAYOUT packet](#)  
[EMR\\_SETLINKEDUFIS packet](#)  
[EMR\\_SETMAPMODE packet](#)  
[EMR\\_SETMAPPERFLAGS packet](#)  
[EMR\\_SETMITERLIMIT packet](#)  
[EMR\\_SETPALETTEENTRIES packet](#)  
[EMR\\_SETPIXELV packet](#)  
[EMR\\_SETPOLYFILLMODE packet](#)  
[EMR\\_SETROP2 packet](#)  
[EMR\\_SETSTRETCHBLTMODE packet](#)  
[EMR\\_SETTEXTALIGN packet](#)  
[EMR\\_SETTEXTJUSTIFICATION packet](#)  
[EMR\\_SETVIEWPORTEXT packet](#)  
[EMR\\_SETVIEWPORTORGEX packet](#)  
[EMR\\_SETWINDOWEXT packet](#)  
[EMR\\_SETWINDOWORGEX packet](#)  
[EMR\\_SETWORLDTRANSFORM packet](#)  
[EMR\\_SMALLTEXTOUT packet](#)  
[EMR\\_STRETCHBLT packet](#)  
[EMR\\_STRETCHDIBITS packet](#)  
[EMR\\_STROKEANDFILLPATH packet](#)  
[EMR\\_STROKEPATH packet](#)  
[EMR\\_TRANSPARENTBLT packet](#)  
[EmrComment enumeration](#)  
[EmrFormat packet](#)  
[EmrText packet](#)  
[Enumeration tables](#)  
[EpsData packet](#)  
[EscapeRecordTypes packet](#)  
**Examples**  
[byte ordering example](#)  
[EMF metafile example](#)  
[EMR\\_BITBLT example \(section 3.2.4, section 3.2.6\)](#)  
[EMR\\_CREATEBRUSHINDIRECT example](#)  
[EMR\\_DELETEOBJECT example \(section 3.2.15, section 3.2.19, section 3.2.20\)](#)  
[EMR\\_EOF example](#)

[EMR\\_EXTCREATEFONTINDIRECTW example \(section 3.2.8, section 3.2.11, section 3.2.13, section 3.2.16\)](#)  
[EMR\\_EXTTEXTOUTW example](#)  
[EMR\\_HEADER example](#)  
[EMR\\_SELECTOBJECT example \(section 3.2.3, section 3.2.5, section 3.2.9, section 3.2.12, section 3.2.14, section 3.2.17, section 3.2.18, section 3.2.21\)](#)  
[EMR\\_SETBKMODE example](#)  
[metafile design examples overview](#)  
[RLE bitmap compression example](#)  
[ExtTextOutOptions enumeration](#)

## F

[FamilyType enumeration](#)  
[Fields - vendor-extensible](#)  
[FloodFill enumeration](#)  
[FormatSignature enumeration](#)

## G

[Glossary](#)  
[GradientFill enumeration](#)  
[GradientRectangle packet](#)  
[GradientTriangle packet](#)  
[GraphicsMode enumeration](#)

## H

[HatchStyle enumeration](#)  
[HeaderExtension1 packet](#)  
[HeaderExtension2 packet](#)

## I

[ICMMode enumeration](#)  
[Illuminant enumeration](#)  
[Informative references](#)  
[Introduction](#)

## L

[Letterform enumeration](#)  
[Localization](#)  
[LogBrushEx packet](#)  
[LogFont packet](#)  
[LogFontEx packet](#)  
[LogFontExDv packet](#)  
[LogFontPanose packet](#)  
[LogPalette packet](#)  
[LogPaletteEntry packet](#)  
[LogPen packet](#)  
[LogPenEx packet](#)

## M

[MapMode enumeration](#)  
[Metafile design examples](#)

[Metafile structure](#)  
[MetafileVersion enumeration](#)  
[MidLine enumeration](#)  
[ModifyWorldTransformMode enumeration](#)  
[MR\\_SETTEXTCOLOR packet](#)

## N

[Normative references](#)

## O

[ObjectCreationRecordTypes packet](#)  
[ObjectManipulationRecordTypes packet](#)  
[Objects](#)  
[OpenGLRecordTypes packet](#)  
[Overview](#)

## P

[Panose packet](#)  
[Path Bracket Record Types](#)  
[PathBracketRecordTypes packet](#)  
[PenStyle enumeration](#)  
[PixelFormatDescriptor packet](#)  
[Point enumeration](#)  
[PolygonFillMode enumeration](#)  
[Proportion enumeration](#)

## R

[RecordType enumeration](#)  
References  
    [informative](#)  
    [normative](#)  
    [overview](#)  
[RegionData packet](#)  
[RegionDataHeader packet](#)  
[RegionMode enumeration](#)  
[Relationship to other protocols](#)  
[RLE bitmap compression example](#)

## S

[Security](#)  
[SerifType enumeration](#)  
[StateRecordTypes packet](#)  
[StockObject enumeration](#)  
[StretchMode enumeration](#)  
[StrokeVariation enumeration](#)  
Structures  
    [Drawing Record Types](#)  
    [enumeration tables](#)  
    [objects](#)  
    [overview](#)  
    [Path Bracket Record Types](#)

## T

[TextAlignmentMode enumeration](#)  
[TransformRecordTypes packet](#)

[TriVertex packet](#)

## U

[UniversalFontId packet](#)

## V

[Vendor-extensible fields](#)  
[Versioning](#)  
[VerticalTextAlignmentMode enumeration](#)

## W

[Weight enumeration](#)  
[Windows behavior](#)

## X

[XForm packet](#)  
[XHeight enumeration](#)